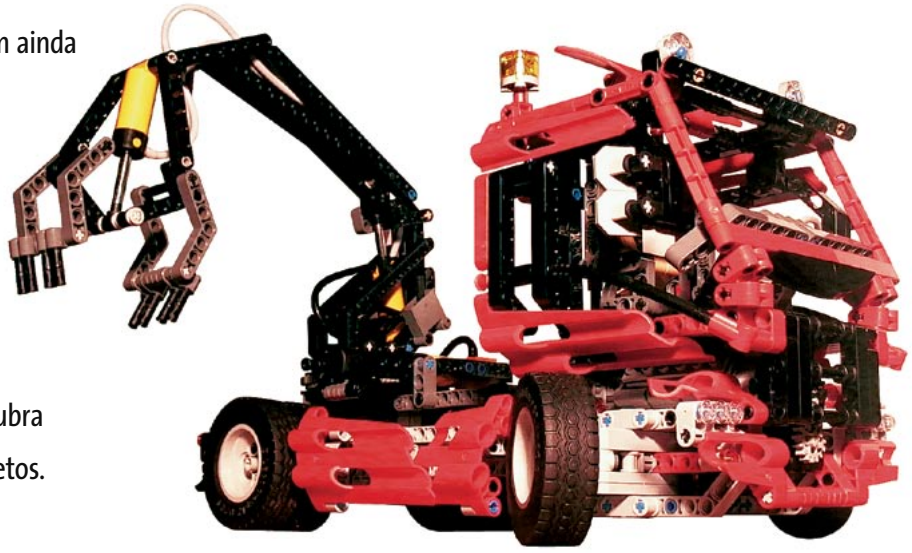
Ambientes de Desenvolvimento e ferramentas para construção e migração de aplicações

Ferramentas poderosas



www.sxc.hu

Letras brancas sobre fundo preto podem ainda ser o ambiente de desenvolvimento preferido dos hackers, mas o Linux não se limita apenas às ferramentas tradicionais. Conheça diversos programas criados para atender às mais variadas expectativas dos desenvolvedores mais exigentes e descubra como eles podem auxiliar em seus projetos.

Por Rodrigo Domingues

Apesar de se ater a uma filosofia simplista, as ferramentas de desenvolvimento presentes nas diversas distribuições Linux possuem recursos suficientes para a produção de aplicações complexas. Porém, para projetos maiores, o uso apenas dos editores de texto, das regras de gerenciamento de compilação e de um ou mais compiladores fazem com que a produtividade geralmente não seja a esperada. Esse tem sido o constante argumento de desenvolvedores de aplicações comerciais para evitar o uso do Linux, preferindo se concentrar em ferramentas mais sofisticadas rodando em sistemas proprietários, capazes de gerenciar múltiplos projetos ou até mesmo de auxiliar na produção de código.

Embora já existam, há algum tempo, ferramentas mais avançadas de gerenciamento de projetos, muitas delas requerem conhecimento de um conjunto de regras de definição, processo que tende a afastar desenvolvedores – estes desejam se concentrar nas tarefas essenciais, ou seja, produzir.

Com a idéia tanto de facilitar o processo de produção quanto de atrair desenvolvedores de aplicações para outros sistemas, contribuidores de diversos projetos de software livre têm unido forças para implementar e disponibilizar ferramentas avançadas que compatibilizem ou compitam com as ferramentas comerciais existentes, tornando o processo de migração mais dinâmico e rápido, conquistando desenvolvedores de todos os tipos de linguagens de programação.

Este artigo apresenta algumas dessas ferramentas, classificando-as quanto ao tipo e linguagem de programação. Descrevemos aqui soluções desde ambientes integrados de desenvolvimento (IDEs) para as linguagens C/C++, Java, Python e Mono a ferramentas RAD (*Rapid Application Development*, ou Desenvolvimento Rápido de Aplicativos) que se utilizam das linguagens *Object Pascal* e *Basic*, em ambientes similares a produtos comerciais como o *Delphi* e o *Visual Basic*.

Anjuta

Desenvolvido para o Gnome utilizando a biblioteca GTK, o *Anjuta* foi um dos primeiros IDEs para o Linux. Com a finalidade de facilitar o desenvolvimento de aplicações em C/C++ (por prover um método automático de configuração da ferramenta *make*, útil para a construção do aplicativo), ele incentivou diversos programadores a auxiliar no desenvolvimento do projeto.

Com um editor de textos integrado e customizável, o Anjuta foi a primeira ferramenta a apresentar o recurso de ocultar ou exibir blocos de código (laços, condições, sub-rotinas e classes) com o intuito de facilitar a leitura de um programa. Sua integração com a ferramenta *Glade*, utilizada para projetar os formulários, janelas e toda a parte visual da interface com o usuário é uma vantagem nas versões atuais. Entretanto, ele ainda não pode ser considerado uma ferramenta RAD, como o *Lazarus* ou o *Gambas*, analisados mais ao final.

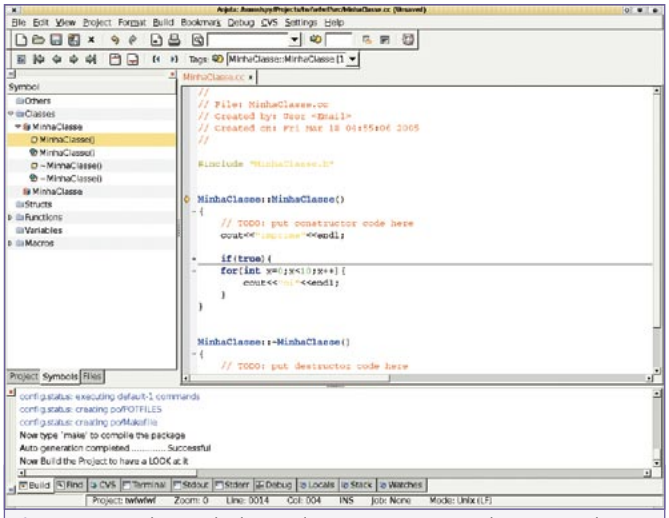


Figura 1: O ambiente de desenvolvimento *Anjuta* e o editor integrado demonstrando a ocultação de códigos e o navegador de classes.

Ele oferece recursos de auto-completar, que permitem que os métodos e atributos de um objeto possam ser consultados em tempo de desenvolvimento sem que o programador necessite recorrer a um manual ou à descrição de uma API, além de recursos de dicas (*hints*) para exibir os atributos necessários para se chamar uma sub-rotina de C/C++ criada no projeto ou nativa do sistema.

Sua capacidade de exibição de estruturas, funções, variáveis, macros e classes presentes no projeto, bem como das descrições de atributos e seus métodos, dentre muitos outros recursos interessantes, são essenciais para facilitar o desenvolvimento de aplicativos complexos.

Sua integração com o popular e poderoso *Gnu Debugger* (GDB) facilita a depuração de aplicações, auxiliando na identificação de erros e no acompanhamento do procedimento de execução da aplicação, já que é capaz de observar os valores de variáveis vigiadas (*watch*).

A curva de aprendizado na utilização do Anjuta é suave, ou seja, uma vez que se está familiarizado com sua interface, a configuração dos projetos e o uso da ferramenta torna-se um processo praticamente trivial, proporcionando um aumento na produtividade e garantindo qualidade no processo de desenvolvimento da aplicação.

Apesar dessa miríade de vantagens, o Anjuta possui a desvantagem de não ter, integrado ao programa, um sistema de documentação sobre o próprio ambiente, sendo necessário recorrer aos documentos externos para poder aprender a operá-lo. Além disso, ele também só é capaz de gerenciar projetos escritos em C ou C++, apesar de realizar marcação de sintaxe (*syntax highlighting*) em programas escritos em outras linguagens, como Pascal, Java ou Perl.

KDevelop

Seguindo a mesma linha de raciocínio que o Anjuta, o grupo responsável pelo ambiente de trabalho KDE lançou o projeto *KDevelop*, que começou na obscuridade mas depois ganhou diversos adeptos devido às suas várias ferramentas.

Assim como o Anjuta, ele possui um editor de código integrado e customizável, recursos para verificação de classes, variáveis, macros e estruturas, ocultação e exibição de blocos de código, depurador integrado e muito mais. Porém, o que torna o KDevelop especial é a sua capacidade de gerenciamento de projetos em diversas linguagens de programação, tais como Python, Java, Ruby, Fortran, Haskell, Perl, Shell Scripts e diversas outras.

O sistema de configuração de projeto para especificação de bibliotecas é ligeiramente mais simples que o do Anjuta devido aos rótulos indicados nas abas. Com uma interface totalmente remodelada a partir da versão 3.0, o KDevelop apresenta um ambiente mais agradável e de fácil utilização, o que não ocorria em versões anteriores.

Duas características interessantes em relação ao KDevelop são a capacidade de reuso de código através dos Snippets, trechos de código rotulados que, com um duplo clique, são adicionados automaticamente ao programa, e os *frameworks* de projetos, que configuram automaticamente as bibliotecas básicas para sua aplicação sem que seja estritamente necessário recorrer à aba de configuração de projeto, caso o projeto não utilize bibliotecas adicionais.

Esses frameworks, como demonstrado na **figura 3**, disponibilizam configurações automáticas para projetos voltados tanto ao KDE quanto ao Gnome, e ainda disponibilizam projetos para sistemas externos, como aplicativos para Windows® (Win32),

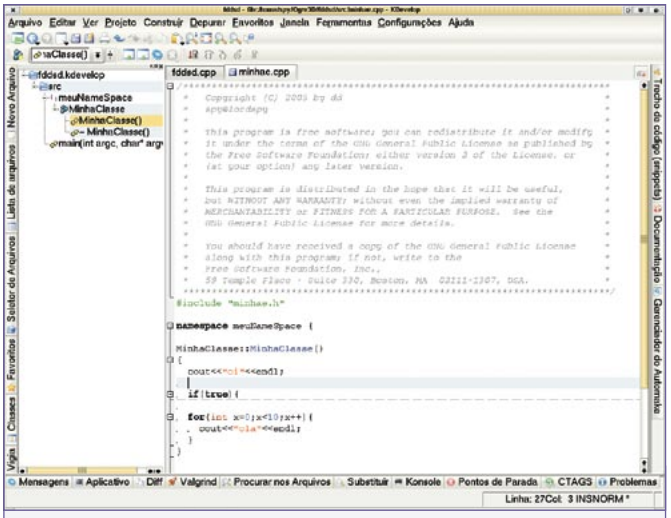


Figura 2: O Ambiente *KDevelop*. Assim como o *Anjuta*, com o editor integrado, demonstra a ocultação de trechos de código e o navegador de classes.

utilizando o *MingW*, e até mesmo para o GameBoy Advance, integrando o emulador *VisualBoy Advance* como ferramenta de execução e verificação do código.

Sua integração com os aplicativos Glade e QtDesigner para a produção de interfaces gráficas (GUI) para aplicações baseadas nas bibliotecas GTK e Qt, respectivamente, é outro fator que pesa bastante a favor dessa IDE. Ela também possui um sistema integrado de documentação de seus recursos, tornando-a uma das ferramentas preferidas dos desenvolvedores; Tudo isso lhe garantiu títulos de "melhor ambiente de desenvolvimento" por três anos consecutivos em várias publicações e premiações internacionais nas áreas de Software Livre ou Open Source.

De acordo com o site oficial, apesar da versão 3.1 possuir apenas a interação com os editores de interface, a versão 3.2.x já possui uma ferramenta RAD para desenvolvimento, nos estilo do Visual Basic ou Delphi, o que pode lhe garantir um quarto ano consecutivo de premiações.

NetBeans

Desenvolvido em Java, o *NetBeans* é um ambiente de desenvolvimento Open Source produzido pela Sun Microsystems que tem como objetivo se tornar uma ferramenta RAD (*Rapid Application Development*) para desenvolvimento de aplicativos Java nos mesmos moldes de seu principal concorrente, o produto proprietário *JBuilder*, da Borland.

Ele possui um sistema de gerenciamento de projetos através da "montagem" de diretórios, como nos sistemas de arquivos tradicionais do Linux, permitindo o desenvolvimento concorrente – ou seja, mais que um programador pode trabalhar simultaneamente em um mesmo arquivo.

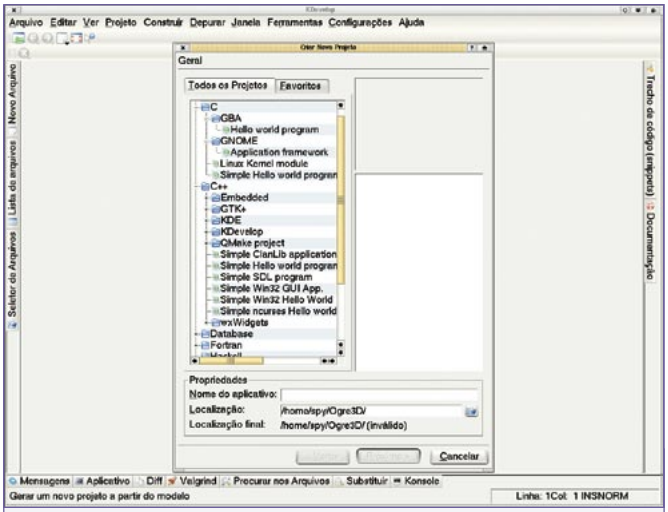


Figura 3: Os diversos projetos já configurados para o KDevelop. A quantidade é razoavelmente impressionante.

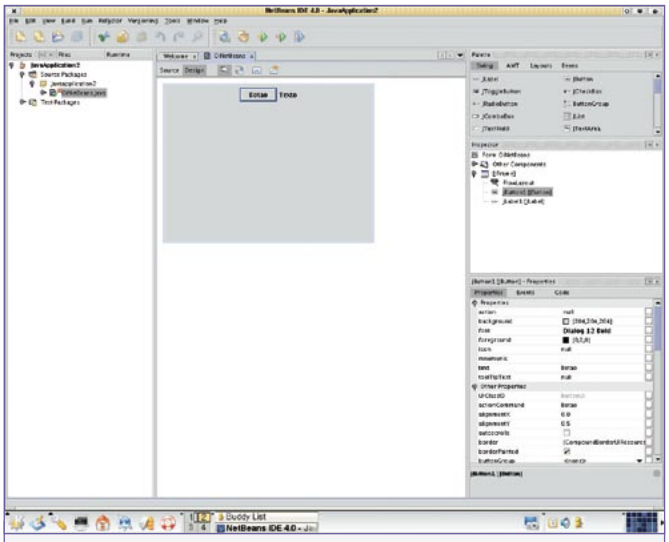


Figura 4: O Editor de Interfaces Integrado, uma facilidade para desenvolvedores de aplicações gráficas.

Seu editor de textos integrado oferece os recursos de completar automaticamente trechos de código, visualizar métodos e atributos de objetos, identação de código e identificação, em tempo de programação, dos trechos de código que possuem erros, o que aumenta a produtividade do programador.

Assim como as IDEs apresentadas anteriormente, o NetBeans também possui visualizador de classes e ferramenta para auxiliar na criação delas; porém, seu diferencial, como demonstrado na **figura 4**, está no editor integrado de interfaces, que gera automaticamente um código limpo e de fácil compreensão para leitores de código em texto puro, caracterizando-o tanto como uma útil ferramenta RAD, quanto como um confiável framework para produção de interfaces.

Além das funcionalidades de edição de código, possui recursos de depuração, desenvolvimento de aplicações web utilizando *Java Server Pages* (*JSP's*) ou *Servlets* através do servidor *Tomcat* e, em uma versão específica, funcionalidades para desenvolvimento de aplicações móveis, além de recursos de análise de código através de técnicas de engenharia de software, ferramentas de banco de dados e de códigos XML. E, por ser desenvolvido em Java, apesar de sua performance poder deixar a desejar quando utilizado em máquinas mais antigas, é uma ferramenta expansível e pode ser encontrada em versões para diversas plataformas.

Eclipse

O projeto *Eclipse* foi iniciado em 1999 pela IBM para servir como um ambiente para produção de ferramentas de desenvolvimento que pudessem ser executadas em diversos sistemas operacionais, tanto em interface gráfica quanto no console,

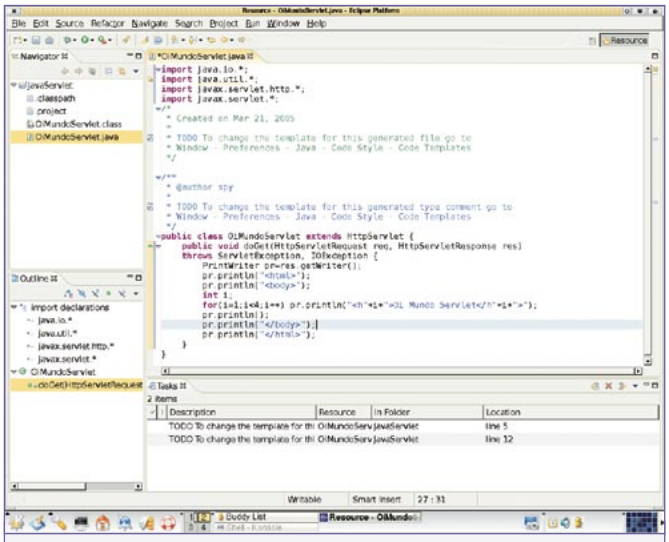


Figura 5: O Ambiente *Eclipse*

utilizando uma filosofia de neutralidade de linguagem, de maneira a permitir criar ferramentas em quaisquer linguagens que se desejasse. Devido a essa filosofia, o projeto tem se tornado uma plataforma universal para a integração de ferramentas de desenvolvimento.

Apesar de trabalhar diretamente com Java, a arquitetura do Eclipse desenvolve o conceito de plugins, nos quais o desenvolvedor implementa a ferramenta necessária para seu projeto e a incorpora ao sistema; isso o caracteriza como expansível e auto-configurável, permitindo, inclusive, o desenvolvimento de plugins proprietários para comercialização.

Sua interface básica, demonstrada na **figura 5**, possui um editor de textos integrado, um visualizador de informações de classes, um visualizador e editor de documentação, um sistema de auxílio à criação de classes e um ambiente de execução Java nativo. Este pode compilar aplicações Java em tempo de desenvolvimento, permitindo identificar erros ou falhas em tempo de programação, assim como no NetBeans.

Atualmente o Eclipse é um projeto Open Source que recebe contribuições de uma grande comunidade dedicada não somente a manter o sistema, mas também a levá-lo a um patamar de qualidade e funcionalidade ainda não alcançado pelas outras ferramentas de desenvolvimento. Entretanto, grande parte do sistema Eclipse é desenvolvido em Java, o que pode levar a problemas de desempenho, principalmente quando executado em máquinas antigas.

MonoDevelop

Um dos mais controversos no movimento Software Livre, o projeto *Mono* está ganhando a confiança dos desenvolvedores. Com o ambiente de desenvolvimento *MonoDevelop*, que se propõe a ser o melhor ambiente de desenvolvimento C# e Mono para ambientes Linux, conquistar desenvolvedores ativos da plataforma *.NET* não será uma tarefa tão árdua quanto se previa.

Ainda em seu estágio inicial, o projeto MonoDevelop está sendo desenvolvido utilizando-se a biblioteca GTK#, uma versão da biblioteca GTK, do Gnome, para o C#. Apesar de ainda não haver uma ferramenta de edição de interfaces com o usuário (entretanto, é possível usar o Glade), há a previsão para a incorporação de uma ao sistema.

Dentre seus recursos, possui um editor de código integrado (veja **figura 6**), é capaz de auto-completar funções nativas e pré-definidas e tem ferramentas de configuração, depuração e gerenciamento de código, além de ferramentas de visualização de informações do projeto, como consulta às classes e rotinas.

Seu sistema de ajuda com documentação tanto para C# quanto GTK# está completo e integrado ao código, proporcionando facilidade de consulta pelo desenvolvedor.

Desenrolando o Python

Para aqueles que querem se iniciar na arte de programação, ou para aqueles que buscam uma ótima ferramenta de scripting para integração com códigos compilados, a linguagem Python tem se provado eficaz em sua tarefa: ser simples, funcional e prover extensibilidade às aplicações.

Com tantas vantagens, essa linguagem não poderia deixar de ter seus ambientes de desenvolvimento. Um deles é a versão 3.0 do editor *Idle*, um ambiente de desenvolvimento bastante funcional para o que a linguagem se propõe. Apesar de simples, o Idle possui marcação de sintaxe, indentação automática e verificação e execução do código em um shell Python integrado.

Para os mais exigentes, existem diversas IDEs que auxiliam no desenvolvimento em Python. O KDevelop é uma boa alternativa; porém, algumas IDEs oferecem recursos adicionais, como por exemplo, a *Boa Constructor* (uma brincadeira com *Boa*

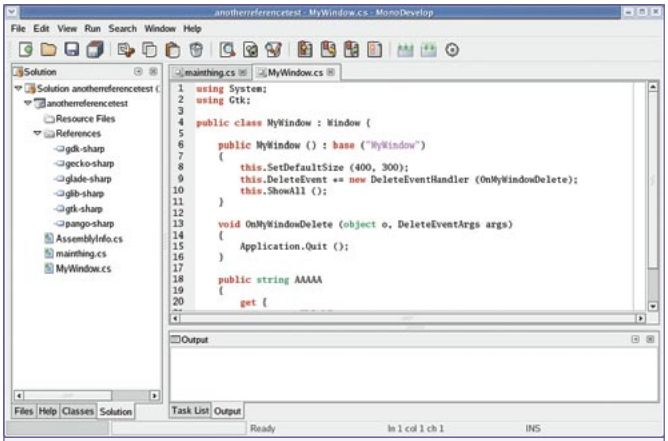


Figura 6: O Ambiente *MonoDevelop*, desenvolvido com *Mono* e *GTK#*

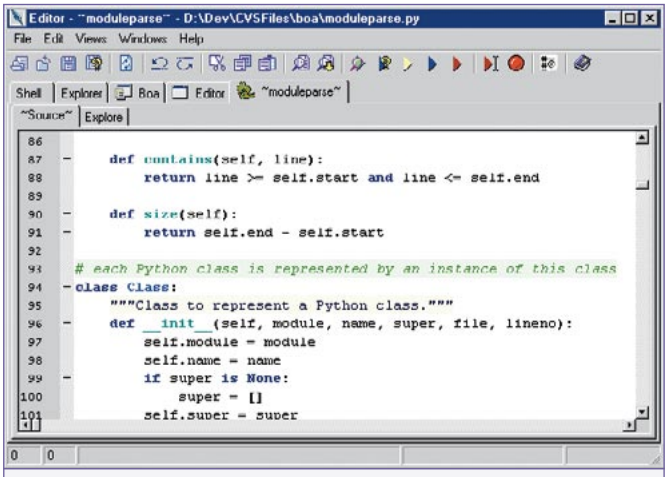


Figura 7: O ambiente de desenvolvimento *Boa Constructor*

Constrictor, família à qual pertencem cobras como as Jibóias e Pítons), demonstrada na **figura 7**, que oferece um visualizador de diagramas de classes importadas (utilizadas), editor de documentação, depurador integrado e editor de interfaces usando WxPython (**figura 8**), dentre outras.

Outra ferramenta bastante comentada dentre os desenvolvedores Python é a *Eric3*, a qual, com exceção do editor de interfaces, possui todas as funcionalidades da Boa Constructor e ainda tem recursos de engenharia de software, tais como um gerador de diagramas UML (**figura 9**), ferramentas integradas para análise de código com a metodologia de *UnitTest* e ferramentas para manutenção ou atualização (*refactoring*) de código.

Lazarus

Quando surgiu no Brasil, a ferramenta RAD *Delphi*, da Borland, tornou-se padrão nacional para desenvolvimento de aplicativos comerciais devido à facilidade de produção da interface com o usuário aliada a uma linguagem de programação de fácil aprendizado, o *Object Pascal*. Nos anos seguintes a mesma Borland lançou o *Kylix*, um ambiente semelhante e compatível com o Delphi para sistemas Linux.

Seguindo a mesma filosofia de facilidade de produção do ambiente gráfico (**figura 10**), aliada à facilidade de programação, o grupo desenvolvedor do *Free Pascal Compiler* disponibilizou o *Lazarus*, um ambiente de desenvolvimento RAD livre e compativel, em quase sua totalidade, com os sistemas desenvolvidos em Delphi.

A ferramenta possui as mesmas características de seu equivalente proprietário: um editor de código integrado, destaque de sintaxe para a linguagem Pascal, sistema de depuração e visualizador de propriedades de classes e componentes. O editor de janelas e formulários e o sistema de criação e inclusão de componentes são bastante atraentes. Com tudo isso, o Lazarus prova que ferramentas livres também possuem qualidade suficiente para o desenvolvimento rápido de aplicações.

Para usuários de Delphi, desenvolver em Lazarus será tarefa trivial: a metodologia de desenvolvimento é a mesma, pois os recursos utilizados na equivalente comercial estão todos disponíveis, como abas contextualizadas de componentes nas categorias de banco de dados, formulários, rede, multimídia etc. Estes componentes são utilizados e programados da mesma maneira que no Delphi.

Gambas

O *Gambas* é um ambiente de desenvolvimento RAD que se propõe a competir com o Visual Basic. Livre, essa plataforma tem sido a preferida pelos desenvolvedores de software de automação comercial para a produção de versões Linux de seus aplicativos.

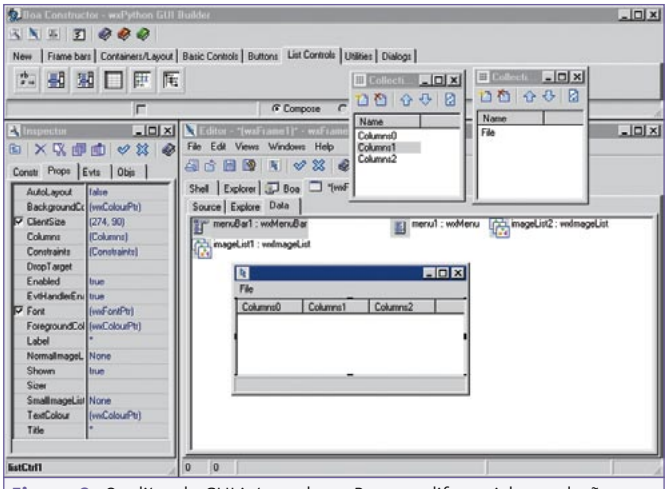


Figura 8: O editor de GUI integrado ao *Boa*, um diferencial em relação a outras IDEs Python

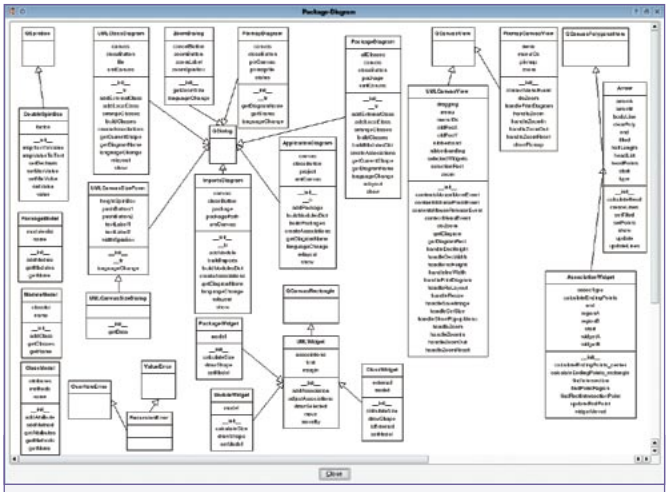


Figura 9: O Visualizador e Gerador de Diagrama UML do Eric3, com recursos de Engenharia de Software

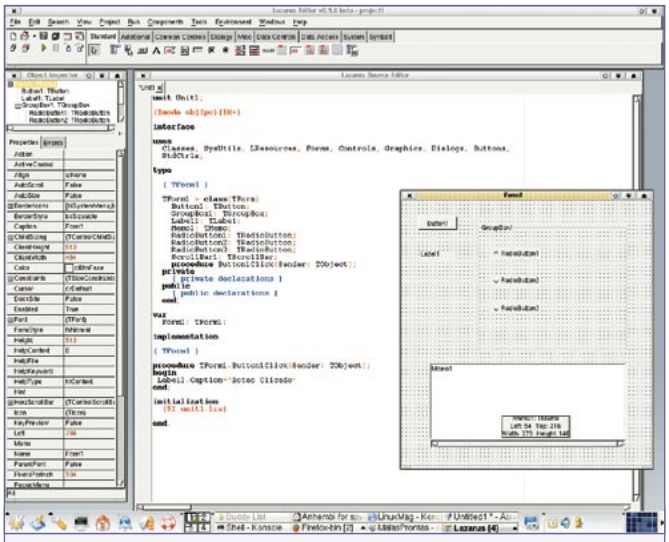


Figura 10: O Ambiente de Desenvolvimento *Lazarus*, liberdade para desenvolvedores *Delphi*.

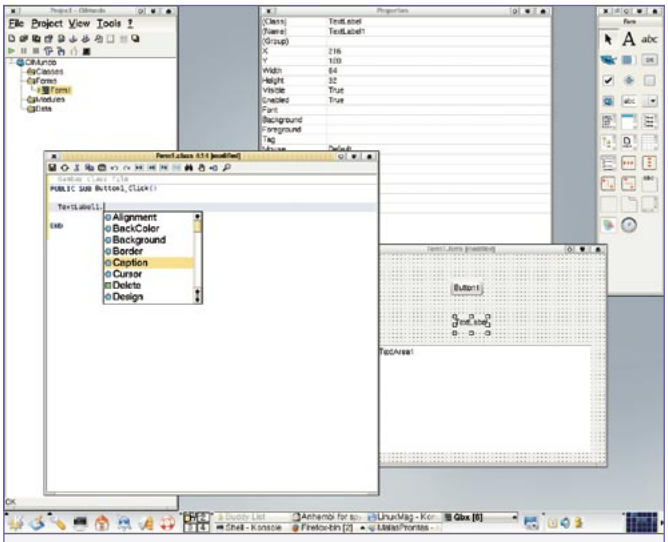


Figura 11: O Ambiente *Gambas* de desenvolvimento. Facilidade de migração aos desenvolvedores acostumados ao *Visual Basic*.

Com uma interface de desenvolvimento semelhante à do Visual Basic (**figura 11**), contendo o editor de textos com o seletor/navegador de funções ou sub-rotinas, o visualizador de propriedades de componentes, a janela principal contendo o navegador do projeto e a janela de formulário, aliadas a uma linguagem essencialmente similar à utilizada na ferramenta proprietária da Microsoft, o Gambas oferece pouquíssima dificuldade de adaptação para os desenvolvedores acostumados com o Visual Basic.

Para produzir uma aplicação simples, como as que os livros básicos apresentam, procede-se de maneira similar à do Visual Basic. Só é necessário ficar atento a alguns pequenos detalhes que, certamente, não causarão problemas, mesmo aos desenvolvedores autodidatas.

Além dos recursos tradicionais, o Gambas possui componentes para acesso e manipulação de banco de dados, recursos multimídia, leitura e escrita a recursos de rede, dentre outros essenciais para o rápido desenvolvimento de uma aplicação de complexidade baixa a média.

Além do Gambas, existem duas outras ferramentas com recursos similares aos disponíveis no Visual Basic. São elas o *hbasic* e o *kbasic*. O último se propõe a ser 100% compatível com a ferramenta da Microsoft, permitindo que aplicações sejam migradas sem dificuldades ou sem a necessidade de alteração de código, apesar de ainda estar em desenvolvimento, mais precisamente na primeira versão beta.

As IDEs e ferramentas de desenvolvimento para Linux tiveram um avanço significativo nos últimos anos. A atenção em manter a metodologia de desenvolvimento ou prover um conjunto de utilitários com funcionalidades similares às das ferramentas comerciais removeu um grande peso da comunidade de desenvolvimento Linux, facilitando também o processo de migração dos programadores. Basta agora apenas quebrar sua inércia, fazendo com que se interessem, conheçam e experimentem essas ótimas ferramentas. ■

Sobre o autor

Rodrigo Domingues (ou *Spy*) é mestre em Computação Gráfica pela Universidade Federal de São Carlos e entusiasta do Linux desde 1997. Atua como consultor, prestando serviços em projetos de pesquisa e desenvolvimento para empresas e laboratórios universitários, é professor no curso de Planejamento e Design de Games nas Universidades Anhembi-Morumbi e se parece com Zaphod Beeblebrox.

Informações

[1] Anjuta anjuta.sourceforge.net
[2] Kdevelop www.kdevelop.org
[3] NetBeans www.netbeans.org
[4] Eclipse www.eclipse.org
[5] MonoDevelop www.monodevelop.com
[6] Python www.python.org
[7] Boa-Constructor boa-constructor.sourceforge.net
[8] Eric3 www.die-offenbachs.de/detlev/eric3.html
[9] Lazarus www.lazarus.freepascal.org
[10] Gambas gambas.sourceforge.net
[11] Hbasic hbasic.sourceforge.net
[12] Kbasic www.kbasic.de