Texto Livre: Linguagem e Tecnologia
E-ISSN: 1983-3652
revista@textolivre.org
Universidade Federal de Minas Gerais
Brasil

Agostinho Rocha, Lucio
Gui Builder Mod: uma ferramenta para criação de aplicações gráficas móveis em Tcl/Tk
Texto Livre: Linguagem e Tecnologia, vol. 11, núm. 3, september-december, 2018, pp. 296-316
Universidade Federal de Minas Gerais

http://periodicos.letras.ufmg.br/index.php/textolivre
Belo Horizonte, v. 11, n. 3, p. 296-316, set.-dez. 2018 – ISSN 1983-3652
DOI: 10.17851/1983-3652.11.3.296-316

GUI BUILDER MOD: UMA FERRAMENTA PARA CRIAÇÃO DE APLICAÇÕES GRÁFICAS MÓVEIS EM TCL/TK

GUI BUILDER MOD: A TOOLKIT FOR BUILDING GRAPHICAL MOBILE APPLICATIONS IN TCL/TK

Lucio Agostinho Rocha
Universidade Tecnológica Federal do Paraná
luciorocha@utfpr.edu.br

RESUMO: O processo de desenvolvimento de interfaces gráficas para o usuário (GUI) de aplicações móveis consome muito tempo durante o ciclo de desenvolvimento. Este artigo tem o objetivo de propor uma nova ferramenta para rápido desenvolvimento de *apps* Tcl/Tk para a plataforma Android. O método utilizado foi criar um *software* em Tcl/Tk para a geração de aplicativos para dispositivos móveis. Essa solução é factível para simplificar o complexo processo de desenvolvimento de novos *apps* e reduzir o esforço inerente de criar aplicações leves que consomem menos recursos de processamento e memória. Como resultado, foi realizado um experimento que mostra o processo de desenvolvimento de uma nova GUI para aplicações robóticas construída com a ferramenta proposta. A principal conclusão é que a ferramenta simplifica a criação de GUIs e gera aplicativos que consumem menos recursos durante sua execução.
PALAVRAS-CHAVE: Android; Tcl/Tk; Interface gráfica do usuário.

ABSTRACT: The development process of graphical interfaces for users (GUI) of mobile aplications consumes a lot of time during the development cicle. This article aims at proposing a new toolkit for fast development of Tcl/Tk apps for the Android platform. The method used was building a Tcl/Tk software to generate aplications for mobile devices. This solution is feasible to simplify the complex development process of new apps, and reducing the inherent effort of create tiny aplications that consumes less resources of processing and memory. As a result was done an experiment that shows the development process of a new GUI for robotic applications built with the proposed toolkit. The main conclusion is that the toolkit simplifies the building of GUIs, and generates aplications that consume less resources during its running.
KEYWORDS: Android; Tcl/Tk; graphical user interface.

1 Introdução

Existem diversas soluções para o desenvolvimento de aplicativos para dispositivos móveis. Em geral, as soluções para a criação de aplicativos utilizam *frameworks*[1] e/ou

1 *Framework* é um *design* reutilizável de todas ou de parte de um sistema que é representado por um conjunto de classes abstratas e pela forma como elas interagem (JOHNSON, 1997). Na prática, um *framework* fornece um conjunto de funcionalidades genéricas para o desenvolvimento de novas aplicações.

http://periodicos.letras.ufmg.br/index.php/textolivre
Belo Horizonte, v. 11, n. 3, p. 296-316, set.-dez. 2018 – ISSN 1983-3652
DOI: 10.17851/1983-3652.11.3.296-316

templates que auxiliam a iniciar a criação de novas aplicações. Aplicações móveis nativas utilizam o *Standard Development Kit* (SDK) fornecido pelo fabricante do dispositivo. As soluções mais conhecidas são Android SDK (para dispositivos Android), Microsoft Visual Studio (para dispositivos Windows Phone), Swift/Objective-C e XCode (para dispositivos iOS). Aplicações nativas têm como principais atrativos o desempenho, acesso aos recursos do *hardware*, e interface gráfica de qualidade, conforme afirma Gasparotto (2018).

Por outro lado, existem *frameworks* híbridos que são geralmente mais leves que as soluções nativas. De acordo com Vasconcellos (2018), o termo híbrido é usado porque parte da aplicação utiliza código nativo para ser distribuído nas lojas de aplicativos, e parte utiliza código não-nativo, como HTML, CSS e JavaScript. A renderização da aplicação é feita em uma *webview* (*browser*), que geralmente possui desempenho reduzido em relação à renderização gráfica nativa. Ainda assim, soluções híbridas são uma alternativa multiplataforma para execução desses aplicativos em diferentes dispositivos móveis. Algumas das soluções híbridas mais conhecidas são o Apache Cordova e o Phonegap. Uma alternativa próxima do ideal é o projeto Xamarin, que utiliza a linguagem C# e também o framework .NET. O projeto Xamarim é usado para criar aplicativos multiplataforma com desempenho próximo ao de soluções nativas. Em razão de existir um complexo processo de desenvolvimento com soluções híbridas, surgiu o projeto Ionic, que é um SDK de componentes visuais em JavaScript para simplificar a criação de aplicações híbridas; o SDK Ionic é baseado no *framework* Angular. Vasconcellos (2018) afirma que o objetivo é buscar oferecer melhor desempenho que outras soluções similares, como jQuery Mobile e Sencha Touch.

Essas alternativas mostram que existe um esforço para o desenvolvimento de ambientes de desenvolvimento leves e robustos, que gerem aplicativos multiplataforma e, de preferência, que tenham ferramentas de desenvolvimento que executem em diferentes sistemas operacionais. Outra motivação é simplificar o complexo processo de configuração do ambiente de desenvolvimento de aplicativos, que envolve a configuração de muitos parâmetros que mudam de *hardware* para *hardware*. Além disso, é fato que grande parte desses *frameworks* hospedam parte de suas dependências em servidores externos e exigem o *download* de muitas dependências para agregar aos aplicativos ao longo do ciclo do processo de criação de aplicativos. Técnicas de compactação e conversão de código, como explicado por Chandrayan (2018), são usadas pela maioria dos *frameworks* nativos e contribuem grandemente para reduzir o tamanho dos aplicativos. Ainda assim, o consumo de recursos de *hardware* é um fator que exige técnicas adequadas do desenvolvedor para otimizar o uso da memória, processador e uso de rede, ainda que com o passar dos anos a quantidade de recursos dos dispositivos móveis tenha crescido enormemente.

Dentre as alternativas para otimizar o desempenho, está o uso de linguagens nativas, como C e C++ embutido em Java com o uso de JNI (*Java Native Interface*) para a execução em plataformas Android. Outro exemplo em especial é a combinação de linguagens Web como HTML, HTML5, CSS, JavaScript e outras para o desenvolvimento de aplicativos multiplataforma, o que auxilia na redução da sobrecarga de execução, mas também exige uma curva de aprendizado elevada para o entendimento de uma intrincada relação entre essas linguagens.

Nesse sentido, o presente artigo apresenta a ferramenta GuibMob versão 3.0. Essa

http://periodicos.letras.ufmg.br/index.php/textolivre
Belo Horizonte, v. 11, n. 3, p. 296-316, set.-dez. 2018 – ISSN 1983-3652
DOI: 10.17851/1983-3652.11.3.296-316

ferramenta é gratuita e de código-fonte aberto, e é usada para otimizar a criação de aplicativos escritos em linguagem Tcl/Tk com GUIs leves e robustas na plataforma Android. A solução proposta é baseada no projeto GUI Builder da ActiveState (ACTIVESTATE, 2018). A principal motivação é oferecer uma alternativa gratuita e de código-fonte aberto para criação de aplicativos Tcl/Tk na plataforma Android e que sejam "leves", ou seja, consumam minimamente os recursos de memória RAM e processador de dispositivos móveis. Além disso, a ferramenta proposta simplifica a criação de interfaces gráficas em Tk com rápida prototipagem.

O restante do artigo está estruturado como segue: a Seção 2 apresenta trabalhos relacionados à criação de aplicativos em Tcl/Tk; a Seção 3 traz a descrição da ferramenta proposta com um experimento que modela uma GUI para controle de robô móvel; a Seção 4 trata da avaliação e resultados de execução, e o detalhamento do processo de compilação e *deploy* (disponibilização do aplicativo no emulador Android), com testes de consumo de RAM e CPU no emulador Android; finalmente, a Seção 5 traz as considerações finais.

2 Trabalhos relacionados

Essa seção descreve os trabalhos relacionados ao processo de desenvolvimento de aplicativos móveis escritos em Tcl/Tk. Atualmente, a ActiveState oferece a IDE Komodo para a criação de interfaces gráficas em Tk (ACTIVESTATE, 2018). Além disso, existem diversas soluções gratuitas e de código-fonte aberto para o desenvolvimento de GUIs Tk no site oficial da linguagem (TCL, 2018; WIKI, 2018a; WIKI, 2018b), mas muitas dessas soluções estão obsoletas e/ou foram descontinuadas. Dentre as soluções mais conhecidas estão o GridPlus (GRIDPLUS2, 2018), Gub (CASSOFF, 2018), SpecTCL (SPECTCL, 2018), Tk GUI Builder (reescrita do projeto SpecTcl, que teve o código-fonte liberado em 2006 e fazia parte do projeto da IDE Komodo), e VisualTcl (ALLEN, 2018).

Desenvolvedores que utilizam a linguagem Tk são capazes de construir GUIs sem o auxílio de *Interface Development Environments* (IDEs), mas a tarefa se torna onerosa com projetos mais elaborados que exigem depuração mais cuidadosa do código. Komodo IDE GUI Builder (ACTIVESTATE, 2018) é a ferramenta mais recomendada para simplificar a criação de GUIs em Tk, mas ainda não possui suporte nativo gratuito e de código-fonte aberto para a geração de código que execute em dispositivos embarcados (KOMODO, 2018). Quanto ao desenvolvimento móvel, a mais conhecida solução é a que utiliza o *framework* AndroWish (ANDROWISH, 2018). Esse *framework* é um SDK para a execução de aplicativos Tcl/Tk em plataforma Android. O projeto AndroWish utiliza os projetos Android SDK e *Native Development Kit* (NDK) para construção de aplicativos Tcl/Tk (NDK, 2018). O projeto Android NDK é designado para incluir código nativo escrito em C e C++ em pacotes da aplicação Android, que é compilado como bibliotecas compartilhadas *Java Native Interface* (JNI). O código Java invoca as funções do código nativo usando JNI. Além disso, o Android NDK é usado para compilar o código em uma biblioteca nativa e empacotá-lo em um arquivo APK. Uma prova de conceito similar ao AndroWish é o projeto undroidwish (UNDROIDWISH, 2018), que usa partes do código do projeto Androwish para execução de código Tcl/Tk em quaisquer outras plataformas móveis.

As linguagens Tcl e Tk são linguagens interpretadas com código-fonte aberto e

http://periodicos.letras.ufmg.br/index.php/textolivre
Belo Horizonte, v. 11, n. 3, p. 296-316, set.-dez. 2018 – ISSN 1983-3652
DOI: 10.17851/1983-3652.11.3.296-316

núcleo escrito em C. Juntas, essas duas linguagens de programação auxiliam a criar aplicações multiplataforma leves e robustas. Mas o uso dessas linguagens em programação para aplicativos móveis ainda é recente, grande parte em função da rápida adoção de soluções nativas e ao avanço dos SDK híbridos com tecnologias Web que simplificam a criação de aplicativos. Ainda assim, é sempre desejável utilizar aplicativos otimizados para a plataforma-alvo, independente da quantidade de recursos do processador embarcado.

Com base no que foi pesquisado, ainda não há uma solução gratuita e de código-fonte aberto que permita criar GUIs em Tcl/Tk integrados em uma única ferramenta. Nesse sentido, a ferramenta proposta GuibMod versão 3.0 (Rocha, 2018) é uma solução gratuita e de código-fonte aberto para a criação de GUIs em Tk para a plataforma Android. Essa ferramenta é baseada no projeto de código-fonte aberto Tk GUI builder (SPECTCL, 2018) e utiliza o *framework* Androwish.

A Tabela 1 mostra uma comparação das mais conhecidas ferramentas de construção de GUIs em Tcl/Tk. É observado que o suporte *mobile* é exclusivo das mais recentes ferramentas e que o código é protegido com restrições de direitos autorais.

Tabela 1: comparação de recentes ferramentas para construção de GUIs Tk.

Ferramenta de construção de GUI para linguagem Tk	**Última atualização**	**Suporte *Mobile***	**Licença**
GuibMod versão 3.0	2018	Sim	GPLv3 / Copyright
AndroWish	2018	Sim	Copyright
Komodo GUI Builder	2018	Sim	Copyright Trial
GridPlus	2015	Não	Copyright
Gub	2018	Não	Copyright
Tk GUI builder	2012	Não	Copyright
VisualTcl	2007	Não	GPLv2
SpecTcl	2002	Não	BSD License

Fonte: autoria própria.

2.1 Desenvolvimento de aplicativos para dispositivos móveis

Com relação ao desenvolvimento de aplicativos para dispositivos móveis, é importante destacar as principais diferenças entre aplicativos nativos e híbridos. A seguir, é feita uma comparação das principais características que diferem esses aplicativos.

a) Aplicativos nativos: utilizam código gerado com o SDK próprio para a plataforma (*hardware* e sistema operacional)-alvo. Vantagens: desempenho; acesso otimizado aos recursos de *hardware*; Interface gráfica de qualidade. Desvantagens: exigente quanto ao *hardware* de desenvolvimento (RAM, CPU, disco, etc.); grande quantidade de dependências de bibliotecas; código do projeto para

http://periodicos.letras.ufmg.br/index.php/textolivre
Belo Horizonte, v. 11, n. 3, p. 296-316, set.-dez. 2018 – ISSN 1983-3652
DOI: 10.17851/1983-3652.11.3.296-316

desenvolvimento dependente do SDK.

b) Aplicativos híbridos: utilizam código nativo para serem distribuídos nas lojas de aplicativos e código de desenvolvimento não-nativo (exemplos: HTML, CSS, JavaScript e outros). Vantagens: menos exigente quanto aos recursos de *hardware* para desenvolvimento; utiliza linguagens abertas e bem conhecidas; multiplataforma. Desvantagens: curva de aprendizado; sintaxe do código muda com novas versões dos *frameworks; d*esempenho próximo ao ideal na plataforma-alvo; não possui acesso otimizado aos recursos do *hardware*; interface gráfica com renderização em *webview*.

Um exemplo bem conhecido de *framework* para criação de aplicativos híbridos é o projeto Apache Cordova. Esse *framework* é um projeto aberto para criação de aplicativos híbridos. Utiliza linguagens Web convencionais (exemplo: HTML, CSS, JavaScript, e outras). Além disso, utiliza o *software* Node.JS como ambiente de execução (*runtime*) JavaScript. Esse *framework* é responsável por fazer o código JavaScript acessar o *hardware* (por exemplo, câmera, acelerômetro, GPS e outros). Apesar de ser possível a criação de todo o aplicativo apenas com esse *framework*, essa tarefa se torna onerosa quanto à parte visual da interface gráfica com o usuário. Para simplificar essa tarefa, é utilizado o Ionic, que é uma versão aberta do *framework* AngularJS. A Figura 1 ilustra uma visão-geral das principais linguagens suportadas e ambientes que suportam os aplicativos gerados pelo Apache Cordova.

Figura 1: apache Cordova para aplicativos móveis multiplataforma.
Fonte: autoria própria.

A Figura 2 ilustra um questionamento sobre as abordagens adotadas na literatura quanto à criação de aplicativos para dispositivos móveis. Nota-se que, à medida que surgem soluções gratuitas e de código-fonte aberto que se propõe como alternativa a soluções proprietárias, aumenta-se consideravelmente o esforço quantitativo para a depuração do código-fonte. Como exemplo, Apache Cordova, Ionic, AngularJS e Node.JS

http://periodicos.letras.ufmg.br/index.php/textolivre
Belo Horizonte, v. 11, n. 3, p. 296-316, set.-dez. 2018 – ISSN 1983-3652
DOI: 10.17851/1983-3652.11.3.296-316

combinam diversas linguagens Web ao invés de utilizarem uma única linguagem para o desenvolvimento.

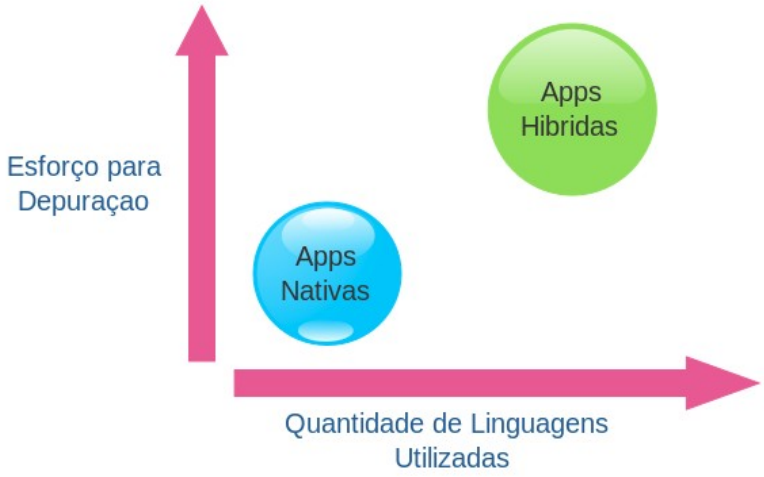


Figura 2: esforço quantitativo para criação de aplicativos.
Fonte: autoria própria.

A dificuldade surge porque as linguagens Web não foram originalmente projetadas para essa finalidade, mas sim adequadas em um emaranhado próprio e não padronizado para tornar disponível para a comunidade que prefere o uso de soluções gratuitas e de código-fonte aberto. Outro fato é que por serem soluções recentes, versões mais recentes desses *frameworks* geram código que não é compatível com as versões anteriores, ou seja, carecem de padronização. Quanto ao desenvolvimento, a documentação de ajuda está em constante mudança e atualmente é preferível assistir a vídeos descritivos do processo. Outro fato é que grande parte do código-fonte utilizado nos aplicativos não é para as funções do núcleo do sistema em si, mas simplesmente para manter o suporte a *plugins* e extensões do *framework*. Ainda assim, é observado que o produto final é visualmente semelhante ao de soluções proprietárias equivalentes, e, por vezes, com desempenho de execução superior em dispositivos móveis de menor custo agregado. De acordo com o site jscrambler (JSCRAMBLER, 2018), os *frameworks* mais utilizados para aplicativos híbridos são:

1) Xamarin (MICROSOFT, 2018): é um esforço da Microsoft para atender à crescente expectativa de usuários de aplicativos Android e IoS por soluções interoperáveis com os dispositivos móveis Windows Phone. Xamarin oferece soluções multiplataforma da *Common Language Infrastructure* (CLI) e *Common Language Specifications*, ou seja, especificações .NET. O código-fonte é escrito em C# e permite escrever código nativo para Android, iOS e Windows Phone, com interfaces nativas e código compatível com plataformas Windows e MacOS (JSCRAMBLER, 2018). Porém, a instalação deste *framework* exige um grande número de bibliotecas proprietárias e a instalação do SDK Microsoft Visual Studio recente com suporte para APIs do sistema operacional Windows 10 (e mais recentes), o que exige um *hardware* recente também.

http://periodicos.letras.ufmg.br/index.php/textolivre
Belo Horizonte, v. 11, n. 3, p. 296-316, set.-dez. 2018 – ISSN 1983-3652
DOI: 10.17851/1983-3652.11.3.296-316

2) PhoneGap (PHONEGAP, 2018): é um *framework* de código-fonte aberto multiplataforma distribuído pela equipe de criação do Apache Cordova. De acordo com o site do projeto, Apache Cordova é a *engine*[2] do *framework*. PhoneGap utiliza linguagens Web HTML, CSS e JavaScript adaptadas para a criação de aplicativos móveis. O núcleo das aplicações utiliza CSS3 e HTML5 para renderização. O acesso às funções do *hardware* é realizado com HTML5 que é embutido em uma *webview* devido a diversas incompatibilidades de *browsers* de dispositivos móveis em relação ao uso de HTML5. Uma grande quantidade de *plugins* pode ser embutida ao ambiente de desenvolvimento, tais como suporte a linguagens TypeScript, *plugins* de acesso a acelerômetro, sistema de arquivos, microfone e outros (JSCRAMBLER, 2018).

3) Intel XDK (INTEL, 2018): O projeto Intel XDK foi descontinuado recentemente, mas possibilitava construir aplicativos multiplataforma para praticamente todas as plataformas de dispositivos móveis e possuía suporte a HTML5, Node.JS, IoT e outros. O site do projeto recomenda o uso de outras alternativas como PhoneGap e Node.JS.

4) Ionic (IONIC, 2018): é um SDK construído acima do Angular.JS e da *engine* Apache Cordova. O *framework* Ionic oferece uma alternativa para criação de GUIs com CSS, HTML5 com apresentação em tempo-real das interfaces gráficas no *browser* do usuário antes do aplicativo ser gerado para o dispositivo móvel.

5) Framework7 (FRAMEWORK7, 2018): é um *framework* HTML gratuito e de código-fonte aberto que gera aplicativos e aplicações Web para iOS e Android. São utilizadas as linguagens HTML, CSS/CSS3 e JavaScript.

6) Appcelerator Titanium (APPCELERATOR, 2018): com uma mistura de ambientes Xamarin e PhoneGap, esse *framework* gera aplicativos multiplataforma para Android e iOS, com código misto nativo e híbrido JavaScript. Appcelerator também utiliza um modelo de desenvolvimento baseado em MVC (*Model View Controller*).

7) Mobile Angular UI (MOBILEANGULAR, 2018): é um *framework* gratuito e de código-fonte aberto que combina as interfaces do Bootstrap versão 3 e AngularJS para criar aplicativos HTML5. De acordo com o site blog.jscrambler.com (JSCRAMBLER, 2018), Angular UI se assemelha a uma extensão do Bootstrap 3, mas sem diversas de suas dependências, e que utiliza diversas diretivas do Angular e outros *scripts* em JavaScript para criar interfaces amigáveis.

8) Onsen UI (ONSEN, 2018): é um *framework* que permite intercambiar o desenvolvimento entre diversos outros *frameworks*, tais como Angular, Angular2, React, Vue.js, Meteor, JavaScript puro ou HTML5 para construção de aplicativos.

9) Sencha Touch (SENCHA, 2018): é uma linha de produtos comerciais para criação de aplicativos móveis com HTML5 e JavaScript. Essa solução fornece uma aparência nativa para as plataformas-alvo iOS, Android, Windows Phone e Blackberry. Oferece um ambiente de desenvolvimento visual com ampla quantidade de bibliotecas.

10) Kendo UI (KENDO, 2018): é um *framework* comercial que usa HTML5 e jQuery para criar aplicativos multiplataforma.

2 *Engine* é o termo usado para indicar um programa que realiza as funções de núcleo para outros programas (TECHTARGET, 2018).

http://periodicos.letras.ufmg.br/index.php/textolivre
Belo Horizonte, v. 11, n. 3, p. 296-316, set.-dez. 2018 – ISSN 1983-3652
DOI: 10.17851/1983-3652.11.3.296-316

11) NativeScript (NATIVESCRIPT, 2018): é um *framework* que utiliza JavaScript e suporta nativamente os *frameworks* Angular e Vue através de *plugins*. O código gerado é nativo para a plataforma-alvo, como se tivesse sido desenvolvido com Android Studio ou Xcode.

12) React Native (REACT, 2018): é um *framework* que permite construir aplicativos móveis com JavaScript. O *framework* gera um aplicativo genérico que posteriormente é formatado para a plataforma-alvo, seja ela Android ou iOS.

3 Descrição da ferramenta proposta

A ferramenta proposta, Guib Mod versão 3.0, é um *software* para criação de aplicativos móveis híbridos. As principais motivações para utilizar esse *software* são:

- Desempenho de execução.
- Reuso de código.
- Reduzir o esforço para criação de GUIs, similar ao de soluções proprietárias.

Figura 3: fluxograma de desenvolvimento de aplicativos com Guib Mob.
Fonte: autoria própria.

Essa ferramenta utiliza o AndroWish SDK com Tcl/Tk 8.6 portado para Android para construir e executar os aplicativos Tcl/Tk. AndroWish é um *framework* para suporte nativo a aplicativos escritos em Tcl/Tk e que executam em plataforma Android 2.3.3 (ou mais recentes) para processadores ARM e x86. A motivação é que aplicativos em Android são escritos em Java e, opcionalmente, com algumas partes em linguagens C e C++. Uma vez que Tcl/Tk é baseado em C, uma camada JNI é usada na interface com Java. O

http://periodicos.letras.ufmg.br/index.php/textolivre
Belo Horizonte, v. 11, n. 3, p. 296-316, set.-dez. 2018 – ISSN 1983-3652
DOI: 10.17851/1983-3652.11.3.296-316

código C é inserido dentro de um projeto Android, com código adicional C de suporte para interagir com Java. A emulação gráfica X11 e manipulação de eventos é feita com SDL 2.0, AGG (*Anti-Grain Geometry*) e fontes FreeType. O código C é pré-compilado para ARM e processadores x86. Finalmente, depende do Android SDK (Build Tools) para gerar o pacote .APK.

Guib Mod versão 3.0 utiliza um *template* pré-compilado do *framework* AndroWish, o que implica que novos projetos não precisam ser inteiramente recompilados (apenas o novo código-fonte gerado precisará ser recarregado). Para desenvolver novos projetos, basta seguir a estrutura de arquivos e diretórios fornecida pelo *template*. Finalmente, o arquivo .APK é gerado com o Android SDK Build Tools através do aplicativo 'bones' do projeto AndroWish (disponível na pasta $GUIB_MOD/tools/bones do *template* do projeto).

O funcionamento da ferramenta proposta é descrito na forma de um experimento para a criação de uma interface gráfica de controle de um robô móvel. A metodologia desse experimento é ilustrada no fluxograma da Figura 3, como segue:

- No passo 1 é feita a criação da interface gráfica em Tk com Guib Mod, o que agiliza a criação através uma interface amigável do tipo WYSWYG (*What You See is What You Get*) com simples arraste dos componentes gráficos Tk, como mostra a Figura 4.
- No passo 2, o projeto deve ser salvo.
- No passo 3 são gerados o código-fonte para execução *standalone* do aplicativo.
- No passo 4 é gerado o código-fonte do *template* Android.
- No passo 5 é necessário que o usuário informe a localização do código Tcl/Tk gerado para inserir na pasta 'assets/app' do novo *template* Android.
- A partir do passo 6, a execução é feita com o AndroWish Surgery, que tem o código incluído no próprio *template*. Ao executar o AndroWish Surgery, o usuário deve informar quais *packages* deseja incluir no seu APK, os arquivos Tcl/Tk que serão incluídos (main.tcl, main_ui.tcl e main.ui), as propriedades do arquivo Manifest.xml, o ícone do aplicativo e a assinatura com o keystore.
- No passo 7 é feita a construção do aplicativo.
- Finalmente, no passo 8, é realizada a execução do aplicativo no dispositivo móvel ou emulador Android, como mostra a Figura 5.

Um vídeo descrevendo o processo está disponível em <https://www.youtube.com/watch?v=PPsoGjHmMJ4>. A geração de todo o código é automática, com mínima intervenção do usuário. Ao criador do aplicativo caberá o papel de preencher o arquvo 'main.tcl' com a ação a ser realizada com a interação do usuário em determinado *widget* (elemento gráfico do pacote Tk), seja essa ação o clicar de um botão, arraste de uma barra de rolagem, validação de preenchimento de campos, ou mesmo uma mensagem de abrir e fechar o aplicativo. Também é papel do criador do aplicativo gerar os certificados digitais válidos para disponibilizar o aplicativo em lojas digitais. Esses certificados podem ser criados com a ferramenta 'bones' disponível dentro da pasta do *template* do aplicativo.

http://periodicos.letras.ufmg.br/index.php/textolivre
Belo Horizonte, v. 11, n. 3, p. 296-316, set.-dez. 2018 – ISSN 1983-3652
DOI: 10.17851/1983-3652.11.3.296-316

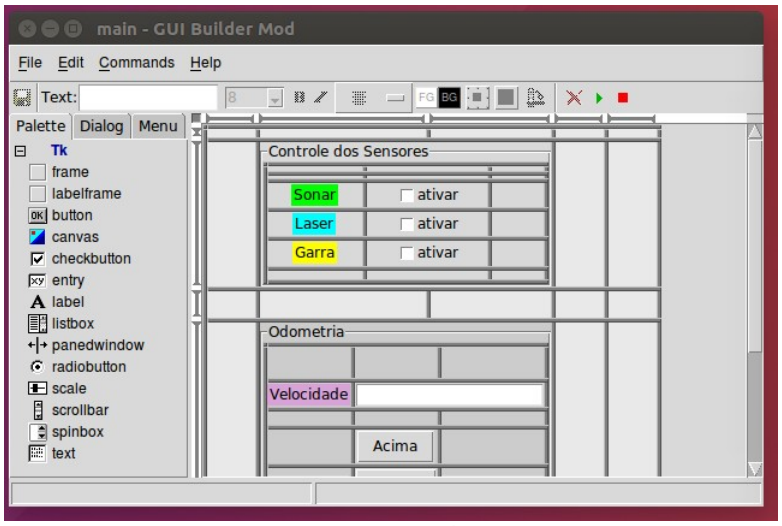


Figura 4: construção da interface gráfica Tcl/Tk com GuibMod.
Fonte: autoria própria.

Figura 5: aplicativo em execução no emulador Android.
Fonte: autoria própria.

3.1 Estrutura de arquivos e diretórios

Para construir aplicativos Tcl/Tk com Guib Mod, é necessário seguir a estrutura de arquivos e diretórios apresentada na Tabela 2.

A Figura 6 ilustra a posição lógica do pacote de *softwares* do Guib Mod versão 3.0 logo acima das bibliotecas do SDK Android. No lado mais à esquerda dessa figura é apresentado o programa *desktop* para criação da interface gráfica (referente à Figura 4), e no lado mais à direita da imagem o mesmo programa em execução em um emulador Android (referente à Figura 5). Portanto, a ferramenta fornece um SDK mínimo para que o usuário crie seus *apps*.

http://periodicos.letras.ufmg.br/index.php/textolivre
Belo Horizonte, v. 11, n. 3, p. 296-316, set.-dez. 2018 – ISSN 1983-3652
DOI: 10.17851/1983-3652.11.3.296-316

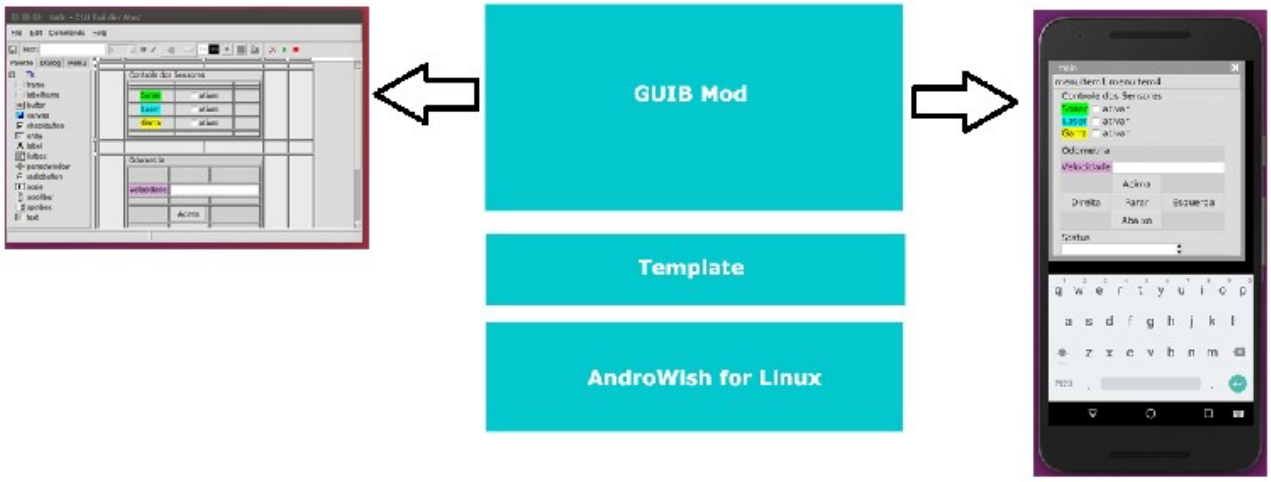


Figura 6: Guib Mod com Android.
Fonte: autoria própria.

Tabela 2: estrutura de arquivos e diretórios do aplicativo.

Conteúdo das pastas	**Descrição do Conteúdo**
$app/assets/app	Código-fonte do aplicativo Tcl/Tk.
$app/bin	Contém o arquivo .apk gerado
$app/gen	Arquivos gerados automaticamente pelo compilador.
$app/libs	Bibliotecas de integração dos scripts com o Android
$app/res	Arquivos de formatação do aplicativo.

Fonte: autoria própria.

Em resumo, o processo de construção é feito como segue: Gui Builder Mod → (main.tcl, main.ui, main_ui.tcl) + geração código-fonte Android *template*→ inclusão dos fontes (main.tcl, main.ui, main_ui.tcl) em $GUIB_MOD/src/new_*template* → Gerar .APK (AndroWish Surgery ou ant debug) → Executar Emulador Android → Instalar .APK (adb install).

O trecho de código da Tabela 3, a seguir, mostra o código-fonte em Tcl/Tk gerado automaticamente pela ferramenta. O próximo passo do usuário é apenas preencher os procedimentos com o código desejado entre as chaves. Por exemplo, para criar uma ação ao clicar em um botão, o usuário deve preencher o trecho entre chaves do procedimento “main::_button_1_command”.

http://periodicos.letras.ufmg.br/index.php/textolivre
Belo Horizonte, v. 11, n. 3, p. 296-316, set.-dez. 2018 – ISSN 1983-3652
DOI: 10.17851/1983-3652.11.3.296-316

Tabela 3: código-fonte do aplicativo gerado com Guib Mod versão 3.0.

```
1   #!/usr/bin/env wish# main.tcl --
2   #
3   # UI generated by GUI Builder Mod Build 1: Educational Version. on 2018-07-12 16:
17:14 from:
4   #    /home/XXXXX/Downloads/main.ui
5   # This file is auto-generated.  Only the code within
6   #    '# BEGIN USER CODE'
7   #    '# END USER CODE'
8   # and code inside the callback subroutines will be round-tripped.
9   # The proc names 'ui' and 'init' are reserved.
10  #
11
12  package require Tk 8.6
13
14  # Declare the namespace for this dialog
15  namespace eval main {}
16
17  # Source the ui file, which must exist
18  set main::SCRIPTDIR [file dirname [info script]]
19  source [file join $main::SCRIPTDIR main_ui.tcl]
20
21  # BEGIN USER CODE
22
23  # END USER CODE
24
25  # BEGIN CALLBACK CODE
26  # ONLY EDIT CODE INSIDE THE PROCS.
27
28  # main::_button_1_command --
29  #
30  # Callback to handle _button_1 widget option -command
31  #
32  # ARGS:
33  #    <NONE>
34  #
35  proc main::_button_1_command args {}
36
37  # main::_button_2_command --
38  #
39  # Callback to handle _button_2 widget option -command
40  #
41  # ARGS:
42  #    <NONE>
43  #
44  proc main::_button_2_command args {}
45
```

http://periodicos.letras.ufmg.br/index.php/textolivre
Belo Horizonte, v. 11, n. 3, p. 296-316, set.-dez. 2018 – ISSN 1983-3652
DOI: 10.17851/1983-3652.11.3.296-316

```
46  # main::_button_3_command --
47  #
48  # Callback to handle _button_3 widget option -command
49  #
50  # ARGS:
51  #    <NONE>
52  #
53  proc main::_button_3_command args {}
54
55  # main::_checkbutton_1_command --
56  #
57  # Callback to handle _checkbutton_1 widget option -command
58  #
59  # ARGS:
60  #    <NONE>
61  #
62  proc main::_checkbutton_1_command args {}
63
64  # main::_entry_1_invalidcommand --
65  #
66  # Callback to handle _entry_1 widget option -invalidcommand
67  #
68  # ARGS:
69  #    <NONE>
70  #
71  proc main::_entry_1_invalidcommand args {}
72
73  # main::_entry_1_validatecommand --
74  #
75  # Callback to handle _entry_1 widget option -validatecommand
76  #
77  # ARGS:
78  #    <NONE>
79  #
80  proc main::_entry_1_validatecommand args {}
81
82  # main::_entry_1_xscrollcommand --
83  #
84  # Callback to handle _entry_1 widget option -xscrollcommand
85  #
86  # ARGS:
87  #    <NONE>
88  #
89  proc main::_entry_1_xscrollcommand args {}
90
91  # END CALLBACK CODE
```

http://periodicos.letras.ufmg.br/index.php/textolivre
Belo Horizonte, v. 11, n. 3, p. 296-316, set.-dez. 2018 – ISSN 1983-3652
DOI: 10.17851/1983-3652.11.3.296-316

```
 92
 93   # main::init --
 94   #
 95   #   Call the optional userinit and initialize the dialog.
 96   #   DO NOT EDIT THIS PROCEDURE.
 97   #
 98   # Arguments:
 99   #   root   the root window to load this dialog into
100    #
101    # Results:
102    #   dialog will be created, or a background error will be thrown
103    #
104    proc main::init {root args} {
105        # Catch this in case the user didn't define it
106        catch {main::userinit}
107        if {[info exists embed_args]} {
108        # we are running in the plugin
109        main::ui $root
110        } elseif {$::argv0 == [info script]} {
111        # we are running in stand-alone mode
112        wm title $root main
113        if {[catch {
114            # Create the UI
115            main::ui  $root
116        } err]} {
117            bgerror $err ; exit 1
118        }
119        }
120        catch {main::run $root}
121    }
122    main::init .
123
```

Fonte: autoria própria.

4 Avaliação e resultados

O objetivo dessa seção é avaliar a viabilidade da criação de aplicativos com a nova ferramenta proposta. Para isso, foram realizados testes quantitativos de consumo de CPU e memória RAM durante a execução de um aplicativo para interação com um robô através de comandos de entrada e saída.

4.1 Análise quantitativa e qualitativa

Foram realizados testes de consumo de RAM e CPU com a ferramenta Android Profiler do Android Visual Studio 3.1.3. O *deploy* e execução foram feitos em um emulador

http://periodicos.letras.ufmg.br/index.php/textolivre
Belo Horizonte, v. 11, n. 3, p. 296-316, set.-dez. 2018 – ISSN 1983-3652
DOI: 10.17851/1983-3652.11.3.296-316

Android, em uma plataforma Linux Ubuntu 16.04 com oito núcleos e 16GB de RAM.

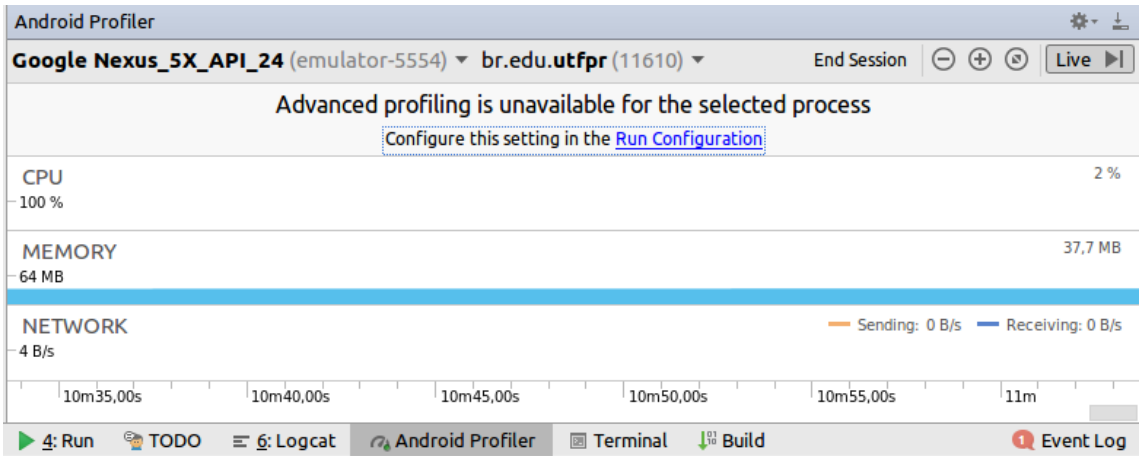
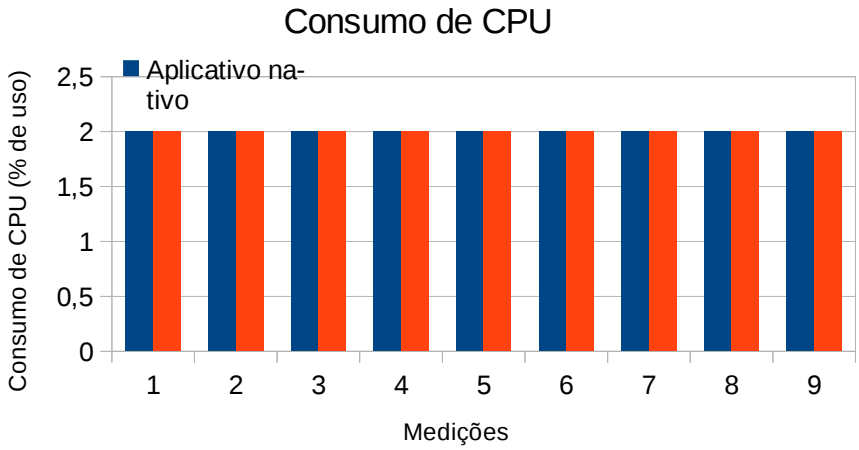


Figura 7: consumo de memória RAM e CPU do aplicativo Guib Mod.
Fonte: autoria própria.

Referente à análise quantitativa, os resultados mostraram um consumo médio em torno de 38MB de RAM, com menos de 2% de uso dos recursos de CPU (Figura 7). Esse resultado é significativo porque valores elevados de consumo poderiam indicar um grande esforço da plataforma-alvo para manter o aplicativo em funcionamento, gerando uma falha de segurança que poderia prejudicar o funcionamento do dispositivo e dos demais aplicativos. Além disso, é comum que *smartphones* recentes tenham pelo menos 2GB de RAM. O consumo apresentado mostra que a execução do aplicativo não prejudica o desempenho das demais aplicações. Uma aplicação similar escrita em Android Studio consome cerca de 10MB de RAM, com menos de 2% de uso de CPU. Ainda assim, o aumento do consumo de RAM é esperado em razão de não ser usado o SDK nativo da plataforma Android para construção do aplicativo, o que pode ter gerado um *overhead* adicional.

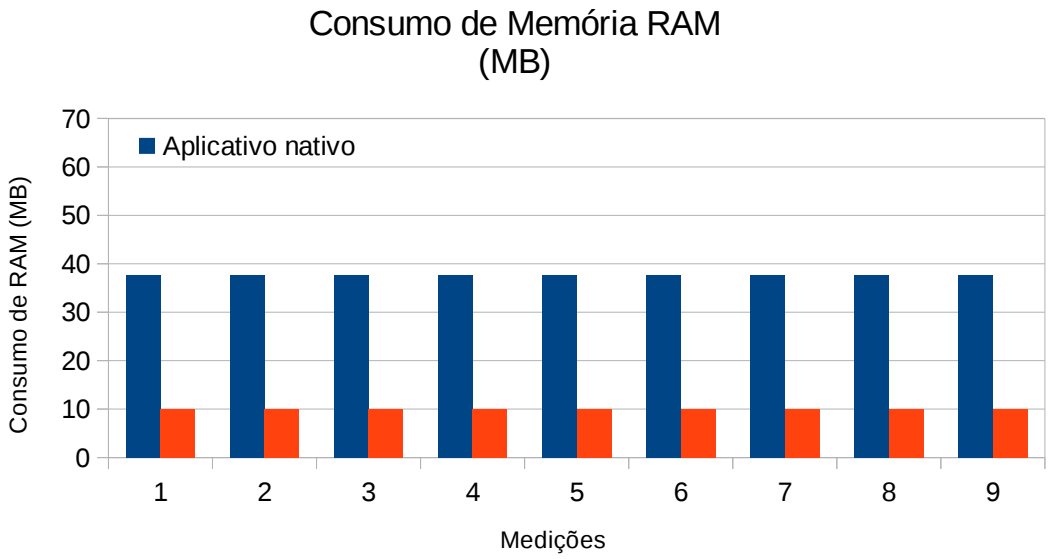
Gráfico 1: consumo de CPU.



Fonte: autoria própria.

http://periodicos.letras.ufmg.br/index.php/textolivre
Belo Horizonte, v. 11, n. 3, p. 296-316, set.-dez. 2018 – ISSN 1983-3652
DOI: 10.17851/1983-3652.11.3.296-316

Gráfico 2: consumo de memória RAM.



Fonte: autoria própria.

O Gráfico 2 também mostra que o consumo de RAM mantém-se razoavelmente constante mesmo durante a interação com o aplicativo em funções básicas de entrada e saída de comandos. O teste não foi exaustivo por se tratar de um aplicativo simples de comandos de entrada e saída para interação com um robô, mas sem a implementação dos comandos. É proposto como trabalho futuro a criação de aplicativos com funcionalidades específicas para avaliar o *overhead* de consumo de recursos em diferentes *smartphones*.

Referente à análise qualitativa, destacamos que os aplicativos gerados com Guib Mod possuem usabilidade e recursos visuais similares à de aplicativos nativos, com desempenho similar e capacidade de realizar as funções de entrada e saída similares à de aplicativos nativos. Porém, é possível melhorar o *look-and-feel* com novas versões da ferramenta proposta. Uma possibilidade de avaliação qualitativa é o oferecimento de um questionário a um grupo de usuários para comparar a usabilidade dos aplicativos com similares nativos. Essa análise é proposta como um trabalho futuro.

4.2 Instalação da ferramenta

Esta subseção apresenta o procedimento de instalação da ferramenta proposta. Os comandos a seguir devem ser realizados em um terminal Bash Linux para realizar a instalação da ferramenta (conforme Tabela 4).

http://periodicos.letras.ufmg.br/index.php/textolivre
Belo Horizonte, v. 11, n. 3, p. 296-316, set.-dez. 2018 – ISSN 1983-3652
DOI: 10.17851/1983-3652.11.3.296-316

Tabela 4: comandos Linux para instalação da ferramenta.

```
$ wget https://dl.google.com/android/android-sdk_r24.4.1-linux.tgz
$ tar xvfz android-sdk_r24.4.1-linux.tgz
//Adquirir os componentes build-tools e platform-tools:
$ android-sdk-linux/tools/android
$ wget https://dl.google.com/android/ndk/android-ndk-r9d-linux-x86_64.tar.bz2
$ tar xvfj android-ndk-r9d-linux-x86_64.tar.bz2
// Configurar as variáveis de ambiente no arquivo $HOME/.bashrc:
export ANDROID_SDK="/home/user/Android/Sdk"
export ANDROID_HOME=$ANDROID_SDK
export PATH=$PATH:$ANDROID_HOME/tools:$ANDROID_HOME/platform-tools:
$ANDROID_HOME
export NDK_PROJECT_PATH=$HOME/Downloads/android-ndk-r9d
export PATH=$PATH:$NDK_PROJECT_PATH
$ source $HOME/.bashrc
//Instalar o pacote de desenvolvimento ActiveTcl da ActiveState:
$ ./activetcl/install.sh
// Fazer o download da ferramenta GuibMod versão 3.0:
$ wget https://sourceforge.net/projects/guibmod/files/latest/download
$ tar xvfz guibmod_v3.tgz
// Usar a ferramenta 'wish' da ActiveState para iniciar o GuibMod versão 3.0:
$ /opt/ActiveTcl-8.6/bin/wish    $GUIB_MOD/src/startup.tcl
// Ao clicar em 'Salvar' será gerada uma nova pasta 'new_template' em
$GUIB_MOD/src
// A pasta 'new_template' contém o código-fonte para criar o aplicativo Tcl/Tk para
Android.
// Após criar a interface gráfica desejada com o Guib Mod versão 3.0 e salvar,
copiar os arquivos gerados
// para a pasta '$GUIB_MOD/new_template/assets/app':
$ cp main.tcl main.ui main_ui.tcl   $GUIB_MOD/src/new_template/assets/app
// Gerar o arquivo .APK do projeto e fazer o deploy no emulador Android (ou
dispositivo Android conectado):
$ /opt/ActiveTcl-8.6/bin/wish    $GUIB_MOD/src/new_template/tools/bones
// (Opcional) Para realizar o deploy do arquivo .APK no emulador Android (ou
dispositivo Android conectado):
$ $ANDROID_SDK/platform-tools/adb -s emulator-5554 install -r
$GUIB_MOD/src/new_template/bin/AplicativoTclTk.apk
```

Fonte: autoria própria.

5 Conclusão

A conexão entre o desenvolvimento de aplicações gráficas leves e intuitivas com o uso de ferramentas como o Guib Mod e o quesito de usabilidade é um campo de pesquisa importante. O modelo de aceitação de tecnologia é fortemente atrelado à percepção dos usuários quanto à facilidade de uso e utilidade, que colaboram para se ter uma atitude

http://periodicos.letras.ufmg.br/index.php/textolivre
Belo Horizonte, v. 11, n. 3, p. 296-316, set.-dez. 2018 – ISSN 1983-3652
DOI: 10.17851/1983-3652.11.3.296-316

favorável no uso da tecnologia (BATISTA; PEDRO, 2015). Nesse sentido, simplificar o processo de criação e modificação de aplicativos para dispositivos móveis é um importante passo para tornar novas ideias de aplicativos viáveis a curto prazo. É ímpar notar que aplicativos são desenvolvidos em um ambiente diferente do local onde eles serão utilizados. Segundo Rahmat et al. (2015), diversos estudos de usabilidade de *software* focam a avaliação da usabilidade de acordo com a eficácia e eficiência dessas aplicações. De maneira similar, a usabilidade de aplicações Web focam a navegação e *links* de acesso. Porém, os estudos sobre a usabilidade de aplicativos focam as restrições do *hardware* que ele executa, ou seja, na forma como são fornecidos os dados para entrada, funcionalidades nativas, desempenho, recebimento/envio de mensagens e aspectos relacionados. Além disso, os autores afirmam que a maioria das avaliações usam métodos heurísticos, de teste ou cognitivos, em parte avaliados através de questionários, *feedback* sobre recursos úteis ou não, e a inspeção de qualidade dos métodos usados para atingir o objetivo proposto pelo aplicativo.

Este artigo apresenta a ferramenta Guib Mod versão 3.0 que é uma solução gratuita e de código-fonte aberto para a criação de GUIs leves e *deploy* de aplicativos Tcl/Tk em plataforma Android. Um vídeo descrevendo o passo a passo do funcionamento da ferramenta é apresentado em <https://www.youtube.com/watch?v=PPsoGjHmMJ4>. É esperado que esse trabalho motive a comunidade a desenvolver novas soluções para a criação de aplicativos leves para plataformas de dispositivos móveis. São propostos como trabalhos futuros o aprimoramento da aparência (*Look-and-Feel)* das GUIs geradas pela ferramenta e suporte para outras plataformas de dispositivos móveis.

As principais vantagens de utilizar a ferramenta proposta são: a) facilidade de criação de interfaces gráficas; b) desempenho dos aplicativos gerados são próximos aos dos aplicativos nativos; c) suporte a linguagens gratuitas para a criação de aplicativos; e d) baixo *overhead* da ferramenta ao ser usada para o desenvolvimento. Porém, apontamos algumas desvantagens: a) ferramenta disponível apenas para plataformas Linux; b) não há um instalador da ferramenta; c) dependente das bibliotecas nativas do Android SDK para geração do aplicativo; e d) dependência de bibliotecas de terceiros para a continuidade do projeto.

Pré-requisitos: a) Uma versão recente do *Java Development Kit*, version 1.8 (ou mais recente); b) Android Standalone SDK tools; c) Android NDK; d) Apache Ant; e e) Tcl/Tk Wish versões 8.5 ou 8.6 (ou mais recente) da ActiveState's ActiveTcl.

Documentação da ferramenta: está disponível uma extensa documentação em Português da ferramenta na pasta $GUIB_MOD/docs.

Referências

ACTIVESTATE. ActiveState – *The Open Source Languages Company*. 2018. Disponível em: <https://www.activestate.com/>. Acesso em: jun. 2018.

ALLEN, S. *Visual Tcl*. 2018. Disponível em: <http://vtcl.sourceforge.net>. Acesso em jun. 2018.

ANDROWISH. *Wiki – The AndroWish Software Development Kit*. 2018. Disponível em:

Belo Horizonte, v. 11, n. 3, p. 296-316, set.-dez. 2018 – ISSN 1983-3652
DOI: 10.17851/1983-3652.11.3.296-316

<http://www.androwish.org/index.html/wiki?name=AndroWish+SDK>. Acesso em: jun. 2018.

ANDROWISH.ORG. *Tcl/Tk on Android*. 2018. Disponível em: <https://androwish.org/index.html/8999948f92969d6b3621412a8d617c5e47139ad4.pdf>. Acesso em: ago. 2018.

APPCELERATOR. *Titanium SDK*. 2018. Disponivel em: <https://www.appcelerator.com/titanium-sdk/4/>. Acesso em: ago. 2018.

BATISTA, S., PEDRO, N. Usabilidade pedagógica: Um fator determinante na adoção do e-learning no ensino superior. In: *2015 10th Iberian Conference on Information Systems and Technologies* (CISTI), Aveiro, 2015, p. 1-4. doi: 10.1109/CISTI.2015.7170452

CASSOFF, S. *Gub – The World's Fastest GUI Builder*. 2018. Disponível em: <https://wiki.tcl.tk/21911>. Acesso em: jun. 2018.

CHANDRAYAN, P. *How your Android code compiles to deliver .APK Package file?* 2018. Disponível em: <https://codeburst.io/how-your-android-code-compiles-to-deliver-apk-package-file-9f180129c9bf>. Acesso em: jun. 2018.

FRAMEWORK7. *Framework7 Documentation*. 2018. Disponível em: <https://framework7.io/docs/>. Acesso em: ago. 2018.

GASPAROTTO, H. M. *Xamarin, Ionic e Cordova*: Conheça o que são e as principais diferenças. 2018. Disponível em: <https://www.devmedia.com.br/xamarin-ionic-e-cordova-conheca-o-que-sao-e-as-principais-diferencas/37690>. Acesso em: jun. 2018.

GRIDPLUS2. *GridPlus2*. 2018. Disponível em: <http://www.satisoft.com/tcltk/gridplus2/>. Acesso em: jun. 2018.

IONIC. *Build Amazing Native Apps in One CodeBase, for any Platform, with the Web*. 2018. Disponível em: <http://ionicframework.com>. Acesso em: ago. 2018.

INTEL. *Intel XDK | Intel Software*. 2018. Disponível em: <https://sotware.intel.com/pt-br/xdk>. Acesso em: ago. 2018.

JOHNSON, R. E. Frameworks = (components + patterns). *Magazine Communications of the ACM*. New York, USA, Vol. 40, Issue 10, p. 39-42, Oct. 1997.

JSCRAMBLER. *12 Framewors for Mobile Hybrid Apps*. 2018. Disponível em: <http://blog.jscrambler.com/10-frameworks-to-mobile-hybrid-apps>. Acesso em: ago. 2018.

KENDO. *React UI Library – Kendo UI – Telerik*. 2018. Disponível em: <https://www.telerik.com/kendo-react-ui>. Acesso em: ago. 2018.

Belo Horizonte, v. 11, n. 3, p. 296-316, set.-dez. 2018 – ISSN 1983-3652
DOI: 10.17851/1983-3652.11.3.296-316

KOMODO. *Creating your first Android app with Cordova and Komodo IDE*. 2018. Disponível em: <https://www.activestate.com/blog/2016/10/creating-your-first-android-app-cordova-and-komodo-ide>. Acesso em: jun. 2018.

MICROSOFT. *Documentação do Xamarin – Xamarin | Microsoft Docs*. 2018. Disponível em: <https://docs.microsoft.com/pt-br/xamarin>. Acesso em: ago. 2018.

MOBILEANGULAR. *Start Learning Mobile Angular UI*. 2018. Disponível em: <https://mobileangularui.com/docs/>. Acesso em: ago. 2018.

NATIVESCRIPT. *NativeScript Documentation*. 2018. Disponível em: <https://docs.nativescript.org>. Acesso em: ago. 2018.

NDK. *Getting Started with the NDK*. 2018. Disponível em <https://developer.android.com/ndk/guides/>. Acesso em: jun. 2018.

ONSEN. *Getting Started*. 2018. Disponivel em: <https://onsen.io/v2/guide>. Acesso em: ago. 2018.

PHONEGAP. *PhoneGap*. 2018. Disponível em: <http://phonegap.com>. Acesso em: ago. 2018.

RAHMAT, H., ZULZALIL, H., GHANI, A. A. ABD and KAMARUDDIN, A. An approach towards development of evaluation framework for usability of smartphone applications. In: *2015 9th Malaysian Software Engineering Conference* (MySEC), Kuala Lumpur, p. 178-182, 2015. doi: 10.1109/MySEC.2015.7475217.

REACT. *React Native – a Framework for Building Native Apps using React*. Disponível em: <https://facebook.github.io/react-native>. Acesso em: ago. 2018.

SENCHA. *Sencha Touch*. 2018. Disponível em: <https://sencha.com/products>. Acesso em: ago. 2018.

SPECTCL. *SpecTcl and GUI Builder Home Page*. 2018. Disponível em: <http://spectcl.sourceforge.net/>. Acesso em: jun 2018.

TECHTARGET. *What is engine?* 2018. Disponível em: <https://whatis.techtarget.com/definition/engine>. Acesso em: nov. 2018.

TCL. *Tcl Developer Xchange*. 2018. Disponível em: <https://www.tcl.tk/>. Acesso em: jun. 2018.

UNDROIDWISH. *undroidwish - Androwish sans the borg, a project just for pun.* Disponível em: <https://www.androwish.org/>. Acesso em: dez. 2018.

VASCONCELLOS, L. *Apps híbridas com Cordova e Ionic*. Disponível em: <https://medium.com/@lfv89/nessa-s%C3%A9rie-divida-em-3-partes-vou-falar-um-pouco-

Belo Horizonte, v. 11, n. 3, p. 296-316, set.-dez. 2018 – ISSN 1983-3652
DOI: 10.17851/1983-3652.11.3.296-316

mais-a-fundo-sobre-desenvolvimento-h%C3%Adbrido-914f22453c83>. Acesso em: jun. 2018.

WIKI. *GUI Building Tools*. 2018a. Disponível em: <https://wiki.tcl.tk/4056?redir=14523>. Acesso em: jun. 2018.

WIKI. *Going Native with Komodo's GUI Builder*. 2018b. Disponível em: <https://wiki.tcl.tk/14522>. Acesso em: jun. 2018.

Recebido em dia 27 de agosto de 2018.
Aprovado em dia 31 de agosto de 2018.