

Introduccion a WxHaskell

San Simon Haskell Hackathone

Carlos Gomez

`carliros.g@gmail.com`

`Abril, 2011`

La Consola

```
[carlos@cathrina src]$ ghci Semantica.hs
GHCi, version 6.12.3: http://www.haskell.org/ghc/
[1 of 3] Compiling Estructura( Estructura.hs, interpreted )
[2 of 3] Compiling Core          ( Core.hs, interpreted )
[3 of 3] Compiling Semantica ( Semantica.hs, interpreted )
Ok, modules loaded: Semantica, Estructura, Core.
```

```
*Semantica> readHLista "../db"
[Nombre_Completo : Daniel Rocha
Universidad : Umss
Correo_Electronico : to.danielrp@gmail.com
```

```
.....
*Semantica> let hlista = it
*Semantica> mostrar nombres hlista
Daniel Rocha
Daniel Saguez Tezanos Pinto
Carlos Gomez
```

```
.....
```

WxHaskell al rescate

WxHaskell nos permite tener
interfaces graficas en Haskell

WxHaskell al rescate

Archivo Lista Help



**Hackathon
Haskell**

1 al 3 de abril del 2011
Centro MEMI - UMSS
Cochabamba - Bolivia

```
trLn . simplex . lines =<< getContents  
] -> [String]  
ecuacion = map word xs  
maxmin = elem (head $ head ecuacion) ["max", "min"]  
maxmin  
en map (\n -> foldl fst n) $ foldl snd n  
se ["No puedo resolver la ecuacion"]  
ing]] -> [(Float, Char)]  
let fo = map (\n -> (fst n * (-1), snd n) (getxy x) --[(-1,x) ,(-1,y)]  
tam = map (\n -> 0.0) [1..(length xs) - 1] --[0,0]
```

Estado DB: Cargado

Data Base File: ./db

WxHaskell al rescate

Archivo Lista Help					
Numero	Nombre	Universidad	EMail	Nivel	
1	Daniel Rocha	Umss	to.danielrp@gmail.com		
2	Daniel Saguez Tezanos Pinto	Mayor de San Simon	danielstp@yahoo.com	Basico	
3	Carlos Gomez	Universidad Mayor de San Simon	carliros.g@gmail.com	Medio	
4	Huanca Balboa Juan Omar	Universidad Mayor de San Simón	oma_2000@hotmail.com	Medio	
5	Rudy Rafael Ramirez Caero	UMSS	mrsblast@gmail.com	Basico	
6	JOHNY ACERO GUAMAN	UINIVERSIDAD MAYOR DE SAN SIMON	johnny_jhonn@hotmail.com	Basico	
7	armin	UMSA	armin@ayni.org	Medio	
8	Haydee Cecilia Piza Conde	Universidad Mayor de San Simon	cecy.ayd@gmail.com	Basico	
9	Jorge cochoza	UMSS	cochozza@hotmail.com		
10	Samuel Herrera	UMSS	thesam7777@hotmail.com	Basico	
11	Mijael Vargas Vargas	Universidad Salesiana de Bolivia	mijacode.007@gmail.com	Medio	
12	Pablo Azero	Universidad Mayor de San Simon		Avanzado	

Numero:	6
Nombre:	JOHNY ACERO GUAMAN
Universidad:	UINIVERSIDAD MAYOR DE SAN SIMON
E-Mail:	johnny_jhonn@hotmail.com
Nivel:	Basico
<div>CancelOk</div>	

¿Que es WxHaskell?

Es una libreria **Grafica** para **Interfaces** de **Usuario (GUI)** para Haskell

Mas detalles de WxHaskell?

- Es portable: unix, linux, window
- Esta en base a **wxWidgets** (version 2.8)
- En linux esta sobre **GTK**
- Su estructura:

WX
wxcore
wxwidgets
~GTK

WxWidgets?

- Libreria GUI para C++
- Portable o multi plataforma
- Varias extensiones: WxHTML, WxDB, Wx...
- ...

WxWidgets + C++ ~ = WxHaskell + Haskell

Hola Mundo WxHaskell

```
module Main where
```

```
import Graphics.UI.WX
```

```
main :: IO ()
```

```
main = start hello
```

```
hello :: IO ()
```


```
hello
```

```
    = do frame [ text := "Titulo Ventana"
```

```
                , layout := floatCenter
```

```
                $ label "Hola Mundo WxHaskell"]
```

```
    return ()
```

A screenshot of a wxHaskell window. The window has a title bar with the text "Titulo Ventana". The main content area of the window is light gray and contains the text "Hola Mundo WxHaskell" in a black, sans-serif font, centered horizontally and vertically.

Que podemos hacer ?

- Podemos hacer varias cosas, ejemplo:
 - Mostrar imágenes, html
 - Usar casi todos sus Widgets Controls

StaticText

Panel

Frame

ScrolledWindow

Dialogs

Events

Variables

Layouts

Button

CheckBox

Choice

ComboBox

ListBox

RadioBox

Spin Control

.....

Un ejemplo Didactico?

Desarrollemos un GUI para administrar
La lista de inscritos al Hackathone.

Los modulos de WxHaskell

- Para usar wxHaskell debes almenos importar el modulo principal de wxHaskell en modulo **Main**:

```
import Graphics.UI.WX
```

- Si quieres puedes importar el Core de WxHaskell para disponer de mas funcionalidad:

```
import Graphics.UI.WXCore
```

La funcion principal y el bucle

- Como cualquier otra libreria GUI, necesitamos un bucle para que nuestro programa no se cierre una ves que lo ejecutamos.

La funcion “`start`” se encarga de iniciar WxHaskell y llevarlo al bucle:

```
main :: IO()
main = start gui
```

Frames de WxHaskell

- Un frame es la ventana padre o principal o root.
- Usaremos un Frame para la ventana principal de nuestro programa:

```
gui :: IO ()  
gui = do f <- frame [text := "Hackathone"]
```

- “frame” es el constructor de Frames, recibe una lista de propiedades.
- La propiedad “text” es el titulo del Frame.

Menus en las Ventanas

- El panel de menus se crea con “menuPane”, donde la propiedad “text” es el nombre del menu.
- Los menus se añaden con la propiedad “menuBar” de Frame.

```
do frame <- frame [text := "Demo"]  
  file <- menuPane [text := "&File"]  
  mclose <- menuItem  
    file  
    [text := "&Close\tCtrl+C"]  
  set frame [menuBar := [file]]
```

Botones en las Ventanas

- Se crean con la funcion “button”
- Reciben la ventana padre donde se colocara.
- Su nombre debe estar en la propiedad “texto” del boton

```
cancel <- button
      panel
      [text := "Cancel"]
ok      <- button
      panel
      [text := "Ok"]
```


Layout de una Ventana

- Podemos acomodar los elementos de un frame de forma:

Manual Usando Layouts

- En la forma **manual** se asignan posiciones
- En la forma de **layout** se utilizan *combinadores* los cuales son asignados a través de la propiedad “layout” de Frame:

```
set f [layout := fill $ widget listBox]
```

- La función “widget” pide el layout de cualquier ventana.

Dialogs de una Aplicacion

- Los dialogs son ventanas prediseñadas.
- Podemos usar para abrir un archivo, seleccionar colores, mostrar mensajes, ...

```
errorDialog parent "error"
```

```
    "fatal error, please re-install windows"
```

```
warningDialog parent "warning" "you need a break"
```

```
fileOpenDialog frame True True "Open image"
```

```
    [("Any file",["*."]), ("Bitmaps",["*.bmp"])] "" ""
```

```
passwordDialog window message caption defaultText
```

```
.....
```

Eventos de wxHaskell

- Todas las ventanas pueden generar eventos:
 - Los botones generan “command”
 - Un click genera “click” o su generico “mouse”
 - Para dibujar se genera “paint”
 - Una tecla presionada genera “key”
 -
- Se pueden capturar los eventos de distintas formas, una de ellas es con la funcion “on”

Eventos de wxHaskell 2

- La función “on” debe estar en la parte de las propiedades de una Ventana.
- Ejemplos:

```
set ventana [on click := return ()]
```

```
mabout <- menuAbout
           mhelp
           [on command := infoDialog f
            "About" about]
```

```
... on command := eliminar f lb dbfile
    db lcItem barraFile barraDB
```

Instalar WxHaskell

- Lo primero, instalar wxWidgets
- Descargar los paquetes de Hackage.
- Forma ~automatica, usando Cabal

```
cabal install wx
```

- Forma Manual

```
runhaskell Setup.hs configure  
runhaskell Setup.hs build  
runhaskell Setup.hs haddock  
runhaskell Setup.hs install
```

Desventajas de WxHaskell (criterio personal)

- No tiene full soporte a wxwidgets
- No full soporte para layouts
- ~Para aplicaciones avanzadas, se necesita conocer wxwidgets, almenos ver la documentacion de wxwidgets y wxhaskell
- Problemas en algunas plataformas (mac, window?)

Ventajas de WxHaskell (criterio personal)

- Buena organizacion, su modelo clases vs objetos
- No todo es IO
- Facil de codificar
- Facil de modificar
- Codigo comprensible
- Tipos de nombre representativos

Referencias

- El código que he utilizado para ciertas partes de la presentación:
<https://github.com/carliros/s2hk>
- Documentación de WxHaskell
- WxHaskell, A Portable and Concise GUI Library for Haskell, Daan Leijen

Mas preguntas?