Tema: IV.
Interfícies gràfics d'usuari

Herramientas Avanzadas para el Desarrollo de Aplicaciones

Departament de Llenguatges i Sistemes Informàtics
Universitat d'Alacant

Contingut

1. Preliminars (I)
2. Preliminars (II)
3. Preliminars (III)
4. MVC (I)
5. MVC (II)
6. MVC: Model. –Capa de l'Aplicació–
7. MVC: Vista. –Capa de Presentació–
8. MVC: Controlador. –Capa d'Interacció–
9. MVC: Diagrama d'interacció entre capes
10. Gtk+ (I)
11. Gtk+ (II)
12. Gtk+ + Vala + signal/handler (I)
13. Gtk+ + Vala + signal/handler (II)
14. Gtk+ + Vala + signal/handler (III)
15. Gtk+ + Vala + signal/handler (IV)
16. Gtk+ Widgets (I)
17. Gtk+ Widgets (II)
18. Glade (I)
19. Glade (II)
20. Glade (III)
21. Glade (IV)
22. Glade (V)
23. Glade + Gtk+ (I)
24. Glade + Gtk+ (II)
25. Glade + Gtk+ (III)
26. Glade + Gtk+ (IV)

Preliminars (I)

- En aquest tema anem a veure com dotar a les nostres aplicacions de un *Interfaz Gràfic d'Usuari* .
- Veurem com es recolza en els conceptes de *programació dirigida per esdeveniments* i *execució diferida de codi* vista en els temes previs.
- Donarem una breu introducció a l'arquitectura MVC ya que per escriure el codi de les nostres aplicacions farem ús de la mateixa.
- L'interfície gràfic de les nostres aplicacions d'escriptori emprarà la biblioteca Gtk+ .

Preliminars (II)

- Veurem com crear gràficament l'interfície de l'aplicació amb un programa dedicat al disseny d'interfícies gràfics de usuari. Aquest es diu Glade .
- Els interfícies gràfics creats amb `glade` seran *carregats* dinàmicament en temps d'execució i mostrats a l'usuari perquè pugui interactuar amb els mateixos.
- Glade genera arxius XML que contenen la descripció del interfície d'usuari dissenyat. Aquests arxius es poden llegir/carregar des de diversos llenguatges de programació: `C`, `C++`, `C#`, `Vala`, `Java`, etc. . .
- Tot això ho farem amb el llenguatge de programació Vala emprat en les pràctiques de l'assignatura per a la part de 'escriptori'.

Preliminars (III)

- Vala empra algunes tecnologies que componen els fonaments de `Gtk+`.
- Aquestes són GLib i Gobject .
- Això influeix perquè amb Vala sigui molt senzill construir aplicacions que tinguin un interfície gràfic d'usuari basat en `Gtk+`.
- Anem a veure un vídeo on en menys de cinc minuts se'ns mostra com crear una aplicació senzilla dotada d'interfície gràfic d'usuari[1].

[1] En aquest cas sense fer ús de Glade.

MVC (I)

- MVC sorgeix juntament amb Smalltalk durant els anys 70.
- És aplicable al desenvolupament de qualsevol aplicació independentment del llenguatge de programació triat.
- No és necessari l'ús d'un llenguatge orientat a objectes para emprar-ho, encara que aquesta metodologia ho faci més senzill.

MVC (II)

- La idea clau de MVC consisteix a dividir el codi d'una aplicació en capes, concretament 3:
  1. **Model**
  2. **Vista**
  3. **Controlador**
- Cadascuna d'aquestes capes pot ser substituïda en qualsevol moment sense afectar a les altres, p.i., tenir diferents vistes per a un mateix model.
- Aquesta divisió del codi garanteix major facilitat de portabilitat i d'adaptació als requeriments de l'usuari.

MVC: Model. –Capa de l'Aplicació–

- És la representació 'programari' del problema a resoldre, els seus dades, funcions, etc. . . – persones, cotxes, seients comptables. . . –
- Proporciona els mètodes necessaris perquè:
  - Es puguin consultar les dades del model ($\approx$ *getters*).
  - Es puguin modificar les dades del model ($\approx$ *setters*).
- Els models no es comuniquen amb les vistes[2], d'est manera aconseguim una major independència entre el codi que constitueix cada capa.
- Un model pot tenir associades diverses vistes.

[2] Encara que en ocasions determinades pot resultar interessant.

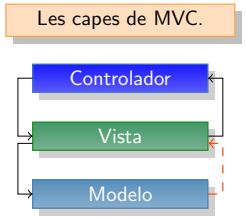
MVC: Vista. –Capa de Presentació–

- Serveix per mostrar a l'usuari les 'dades' del model que li interessen en cada cas –el nom d'una persona, la velocitat de un cotxe, l'import d'un seient. . . –
- Una vista no té perquè ser solament en manera gràfica, pugues ser en manera text. . .
- Les vistes es comuniquen amb els models de forma bi-direccional –sol·liciten informació, poden modificar informació–
- En l'arquitectura MVC original les vistes es poden 'niar' donant lloc al que es diu una vista principal –*top-view*– composta de subvistas. A l'efecte de la assignatura només usarem vistes simples, no compostes per unes altres.

MVC: Controlador. –Capa d'Interacció–

- Conté el codi que fa d'interfície entre els dispositius d'entrada –teclat, ratolí, etc. . . – i les capes de Vistes i Models.
- Aquest és el codi que permet a l'usuari interactuar amb les Vistes.
- Normalment no haurem d'escriure codi relacionat amb aquesta capa ja que el codi que aniria aquí és el que proporciona la biblioteca gràfica utilitzada (Gtk+ en el nostre cas).

MVC: Diagrama d'interacció entre capes



La connexió entre el model i les vistes associades només té sentit si les dades del model es van a modificar internament per algun càlcul i volem que s'actualitzin les vistes associades a aquest model.

Gtk+ (I)

- Gtk+ es desenvolupa com toolkit lliure para l'aplicació gimp de tractament de imatges. Avui dia és una de les bases de l'escriptori gnome .
- Es distribueix amb llicència LGPL.
- Disposem d'una extensa documentació para Gtk+ i per usar-ho des de Vala en format electrònic que es pot consultar en línia.
- També consta d'un constructor gràfic de l'interfície de la aplicació: glade .

Gtk+ (II)

- Gtk+ és actualitzat sistemàticament un parell de vegades a l'any, avui dia podem trobar-nos amb les versions 2.$x.i$ (en manera manteniment) i, l'actualment activa, 3.$x.i$. En el laboratori de pràctiques de la EPS tenim instal·lat Gtk+ 3.$x.y$.
- Com veurem més endavant, això influeix a l'hora de les opcions que hem de donar al compilador de Vala: `--pkg gtk+-2.0` o `--pkg gtk+-3.0`.
- El que denominem de forma general Gtk+ és un compendio de una sèrie de biblioteques: Glib, GdkPixbuf, Gdk, Gtk, Atk y Pango.
- L'estructura interna de Gtk+ és la d'una jerarquia de classes formada per diversos arbres (diferents arrels) amb herència simple.
- Aquests arbres representen a cadascuna de les biblioteques que hem comentat abans (glib, gdk, gtk, etc...).

Gtk+ + Vala + signal/handler (I)

- L'ús de Gtk+ des de Vala es basa en el que hem vist en els temes anteriors: esdeveniments/assenyalis i manejadores/callbacks.
- Els elements d'interfície d'usuari (widgets, controls) que proporciona Gtk+ defineixen una sèrie de senyals que poden emetre.
- Nosaltres ens dediquem a connectar-los els mètodes o funcions del nostre codi que fan de manejador o callback, p.e. consultant la documentació de la classe Button trobem un apartat dedicat a senyals:

```
1   public virtual signal void activate ()

3   The activate signal on GtkButton is an action signal and
    emitting it causes the button to animate press then
5   release....

7   public virtual signal void clicked ()

9   Emitted when the button has been activated (pressed and
    released).
11
    ...
```


Gtk+ + Vala + signal/handler (II)

Vegem un exemple complet:

```
    // File: gtk-hello.vala
2   using Gtk;

4   int main (string[] args) {
      Gtk.init (ref args);
6
      var window = new Window (); //Gtk.Window (using Gtk)
8     window.title = "First GTK+ Program";
      window.border_width = 10;
10    window.window_position = WindowPosition.CENTER;
      window.set_default_size (350, 70);
12    window.destroy.connect (Gtk.main_quit);

14    var button = new Button.with_label ("Click me!");
      button.clicked.connect (() => {button.label = "Thank you";});
16
      window.add (button);
18    window.show_all ();

20    Gtk.main ();
      return 0;
22  }
```

Se compila así: `valac --pkg gtk+-3.0 gtk-hello.vala`.

Gtk+ + Vala + signal/handler (III)

Destacar del codi anterior:

- using Gtk: equivalent a un `import` de Java o `#include` de C o C++ per tenir accés a declaracions/definicions de Gtk+.
- Gtk.init (ref args): Inicia la biblioteca Gtk+. Es indispensable fer-ho sempre al principi del programa principal.
- var window = new Window (): Crea un objecte de classe `Gtk.Window`, és a dir, una finestra sobre la qual poder afegir altres elements d'interfície d'usuari.
- var button = new Button.with_label("Click me!"): Crea un objecte de classe `Gtk.Button`.

Gtk+ + Vala + signal/handler (IV)

- button.clicked.connect(() => {button.label = "Thank you";}): La classe `Button` disposa del senyal `clicked`, aquí li connectem un manejador. En aquest cas és una funció-λ.
- window.add (button): Afegim el botó creat a la finestra.
- window.xou_all(): La finestra fa visibles tots els widgets que contingui.
- Gtk.main(): És el bucle d'espera d'esdeveniments, d'ell solament sortim per finalitzar l'aplicació.

Gtk+ Widgets (I)

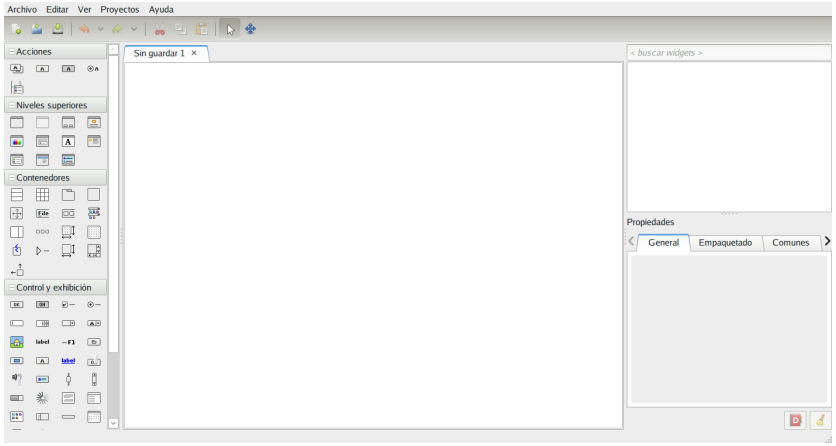
- Gtk+ ens ofereix una col·lecció important de widgets predefinits organitzats com una jerarquia de classes amb herència simple. Podem veure-la aquí
- La classe basi de qualsevol *element d'interfície de usuari* és la classe GtkWidget .
- En Vala el prefix `Gtk` de qualsevol identificador, p.e. 'Gtkwidget' s'interpreta com un *espai de noms*, per la qual cosa l'identificador en Vala sería: 'Gtk.Widget'.
- Disposem de la documentació equivalent per a l'adaptació de Gtk+ a Vala aquí .

Gtk+ Widgets (II)

- En Gtk+ un widget normalment solament pot contenir a un altre widget.
- Per crear interfícies d'usuari funcionals necessitem solucionar aquesta limitació.
- Existeix un tipus especial de widgets que són els *contenidors*, concretament les classes derivades de Gtk.Box . Pots veure més informació sobre les seves classes derivades aquí y en Gtk.HBox y Gtk.VBox .
- També és aconsellable que coneguis el *contenidor en forma de taula*: Gtk.Table . Tens més información sobre ell en la documentació de Gtk.
- Visualment no tenen cap aparença però tenen com característica que poden contenir més d'un widget, així com gestionar l'espai que ocupen i quin ocorre quan canvia el grandària d'aquest espai.

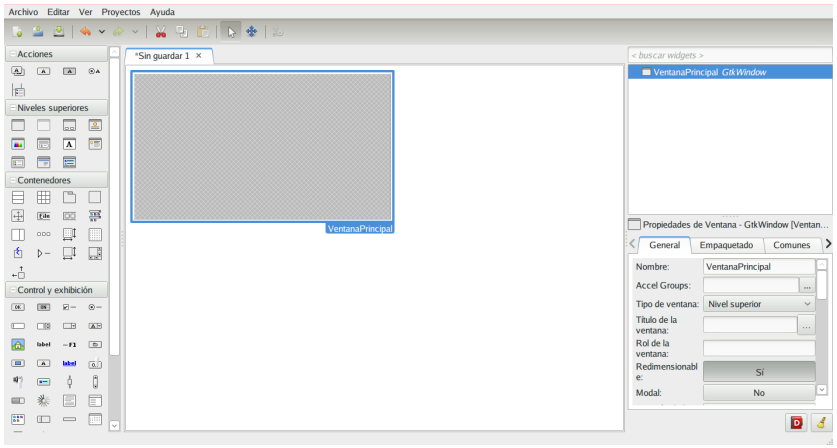
Glade (I)

- Glade és el constructor gràfic oficial d'interfícies d'usuari per Gtk+.
- L'aspecte que presenta és així:



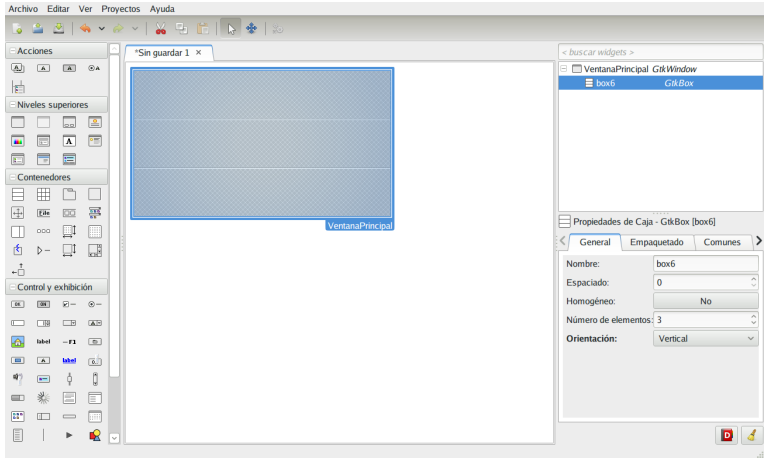
Glade (II)

Creguem les finestres, diàlegs, etc... inicials triant-los d'entri la llista de 'Nivells superiors':



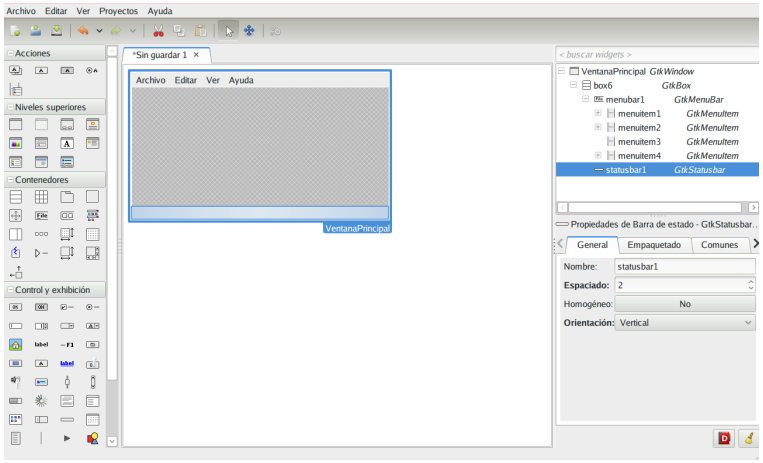
Glade (III)

Li afegim els 'contenidors' necessaris per construir nostre interfaz:



Glade (IV)

Inserim en els 'buits' dels 'contenidors' els widgets que necessitem, p.i. botons, etiquetes de text, camps de text editable, etc... :



Glade (V)

- Una vegada tenim creat l'interfície ho guardem des del menú 'Arxiu'.
- Són arxius de text en format `xml`.
- Solen portar l'extensió '`.ui`'.
- Des de la nostra aplicació els carreguem dinàmicament amb un objecte de la classe '`Gtk.Builder`'.
- Amb el mètode '`add_from_file`' llegim l'arxiu '`.ui`'.
- Amb el mètode '`get_object`' carreguem un a un pel seu nomeni els widgets que ens interessen de l'arxiu '`.ui`'

Glade + Gtk+ (I)

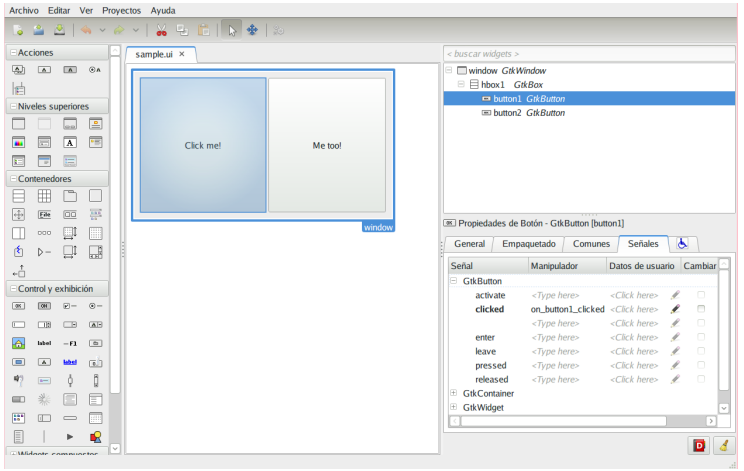
- Vegem com podria quedar un exemple de codi similar al vist en la transparència 15.

```
// valac --pkg gtk+-3.0 --pkg gmodule-2.0 gtk-builder-sample.vala
using Gtk;
public void on_button1_clicked (Button source) {
  source.label = "Thank you!";
}
public void on_button2_clicked (Button source) {
  source.label = "Thanks!";
}
int main (string[] args) {
  Gtk.init (ref args);
  try {
    var builder = new Builder ();   // cargador de archivos de Glade
    builder.add_from_file ("sample.ui");  // carga del interfaz
    builder.connect_signals (null);       // auto conexion de senyales
    var window = builder.get_object ("window") as Window;
    window.show_all ();
    Gtk.main ();
  } catch (Error e) {
    stderr.printf ("Could not load UI: %s\n", e.message);
    return 1;
  }
  return 0;
}
```

L'interfície d'usuari ho podem descarregar de sample.ui

Glade + Gtk+ (II)

Vist `sample.ui` des de Glade té aquest aspecte:



Glade + Gtk+ (III)

- Podem connectar mètodes com manejadores de senyals.
- En aquest cas cal seguir unes normes per donar noms als senyals en Glade.
- Veamoslo amb un exemple:

```
using Gtk;

namespace Foo {
  public class MyBar {

    [CCode (instance_pos = -1)]
    public void on_button1_clicked (Button source) {
      source.label = "Thank you!";
    }

    [CCode (instance_pos = -1)]
    public void on_button2_clicked (Button source) {
      source.label = "Thanks!";
    }
  }
}

// ...
var object = new Foo.MyBar ();
builder.connect_signals (object);
// ...
```


Glade + Gtk+ (IV)

- Si declarem els mètodes que faran de callbacks dins d'una classe i/o dins d'un espai de noms...
- ... en Glade haurem de precedir el nom del mètode que farà de callback amb el nom de l'espai de noms i/o la classe a la qual pertany, en minúscules i separats per símbols de subratllat.
- Per exemple: '`Foo.mybar.on_button1_clicked`' en Glade seria: '`foo_my_bar_on_button1_clicked`', com podem veure en aquesta imatge: