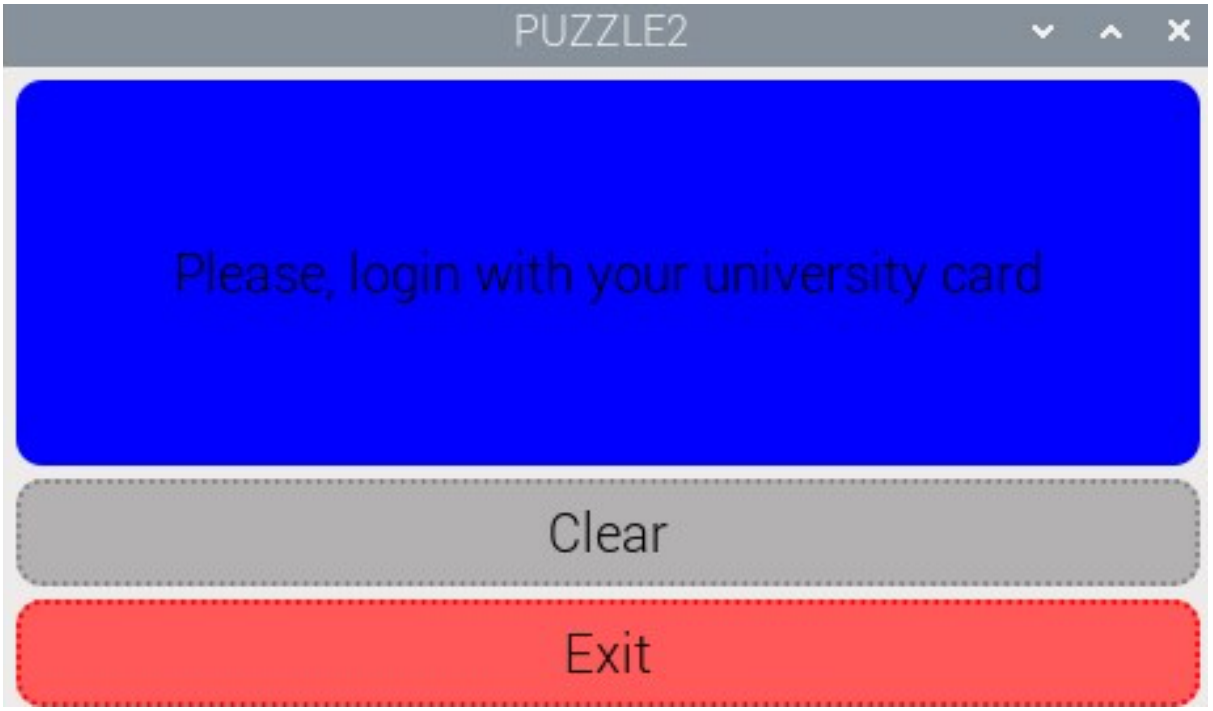
MEMÒRIA PUZZLE 2

Interfície gràfica



Hugo Roldan Lopez

ENUNCIAT

Al puzzle 2 es demana crear una interfície gràfica del puzzle 1 amb la biblioteca PyGObject.

PROCEDIMENT

Aquesta biblioteca ens permet utilitzar Gtk per crear la nostra interfície d'usuari (GUI) i tots els seus elements. Gtk és una biblioteca que està escrita en C, per tant necessitem un *binding* que ens permeti utilizar-la a Python. Per instalar PyOBject executo la següent comanda a la terminal:

- ***sudo apt install python3-gi***

Aquest paquet és el que ens permet incloure les llibreries Gtk, GLib y Gdk amb la següent línia de codi:

from ***gi.repository*** *import* ***Gtk, GLib, Gdk***

Ón Gtk és la biblioteca principal que permet crear els *widgets*, interactuar amb el usuari i en general fer la estructura de la interfície. Després **GLib** és la llibreria que permet entre altres funcions gestionar la interacció de fils secundaris amb la interfície, això ens permetrà fer servir el mètode *read_uid* del puzzle 1 sense tenir problemes amb l'execució principal. **Gdk** és el motor gràfic de Gtk, ens permet entre altres funcions cambiar el fons dels widgets de forma dinàmica.
Les tres llibreries estan escrites en C, per tant instalar PyGObject permet utilitzar-les a Python.

Seguidament, instalo el paquet gir1.2-gtk-3.0:

- ***sudo apt install gir1.2-gtk-3.0***

El paquet conté els arxius GIR (GObject Introspection Repository) per Gtk3. Aquests arxius permeten al paquet instalat prèviament **python3-gi** conèixer les funcions de Gtk en C i utilitzar-les en Python. Transforma els arxius .c en arxius llegibles per Python.
Tots els paquets instal·lats a diferència del puzzle 1 s'instal·len des de el gestor de paquets del sistema APT.

CODI

He volgut dividir el codi en tres classes, una per l'aplicació, que activa la finestra, la finestra i una classe anomenada *widgetManager.* La idea d'aquesta última és actuar com la classe que permet a la finestra realitzar les acciones relacionades amb els *widgets.* Per tant si a la classe finestra vols crear o editar un *widget,* ho faràs a través de la classe *widgetManager.* El meu objectiu amb això és tenir un codi modular, on es dóna a entendre que la classe finestra, es on es troben tots els elements que afegim, però aquests són gestionats a través d'una altre classe.
Per estilitzar els *widgets* he utilitzat el llenguatge CSS, que em permet afegir als *widgets* marges, radi de curvatura, color del fons, tamany del text… Aquesta tècnica permet presentar de forma clara com s'ha configurat cada *widget.*

GTK no permet que s'executin mètodes bloquejants, (congelaria la interfície) que es justament el cas del nostre mètode ***read_uid()*** del puzzle 1. Per solucionar això, creo un fil secundari que s'encarrega de llegir la uid, executant aquest mètode sense afectar la interfície.
Per tant, un cop el fil auxiliar obté la uid, utilitza el mètode ***GLib.idle_add()*** de la librería GLib. Aquest mètode permet passar per argument una funció juntament amb els paràmetres d'aquesta. El que fa aquesta línia de codi és fer que la funció passada per argument, s'executi al fil principal de la interfície. D'aquesta manera podem obtenir la uid al nostre *thread* principal i mostrar-la a la finestra.
El codi es troba a continuació. El codi amb millor format i amb comentaris es troba al meu Github.

Python

```python
import gi
import puzzle1
import threading
gi.require_version("Gtk", "3.0")
from gi.repository import Gtk, GLib, Gdk

WELCOME_STRING = "Please, login with your university card"
GREEN_COLOR = Gdk.RGBA(0.0, 1.0, 0.0, 1.0)
BLUE_COLOR = Gdk.RGBA(0.0, 0, 1, 1.0)



class MyWindow(Gtk.ApplicationWindow):

    def __init__(self,widgetManager):
        super().__init__()
        self.wM = widgetManager
        self.myReader = puzzle1.Rfid_522()

    def configure_window(self,amplada,altura,posició,titol):
        self.set_title(titol)
        self.set_default_size(amplada,altura)
        self.set_position(posició)
```

```
    def start_boxes(self):
        self.main_box = self.wM.create_box(Gtk.Orientation.VERTICAL,0)
        self.add(self.main_box)

    def start_labels(self):
        self.welcome_label = self.wM.create_label(WELCOME_STRING)
self.wM.add_widget_box_start(self.main_box,self.welcome_label, False, False,0)
self.wM.set_widget_name(self.welcome_label,"welcome_label")

    def start_buttons(self):
        self.exit_button = self.wM.create_button("Exit")
        self.exit_button.connect("clicked",Gtk.main_quit)
self.wM.add_widget_box_end(self.main_box,self.exit_button, False, False, 0)
self.wM.set_widget_name(self.exit_button,"exit_button")

        self.clear_button = self.wM.create_button("Clear")
self.clear_button.connect("clicked",self.reset_window)
self.wM.add_widget_box_end(self.main_box,self.clear_button, False, False,0)
self.wM.set_widget_name(self.clear_button,"clear_button")

    def start_reading_thread(self):
        self.thread = threading.Thread(target=self.rf_reading_task)
        self.thread.daemon = True
        self.thread.start()

    def rf_reading_task(self):
        self.myReader.read_uid()
        GLib.idle_add(self.update_window, self.myReader.uid)
```

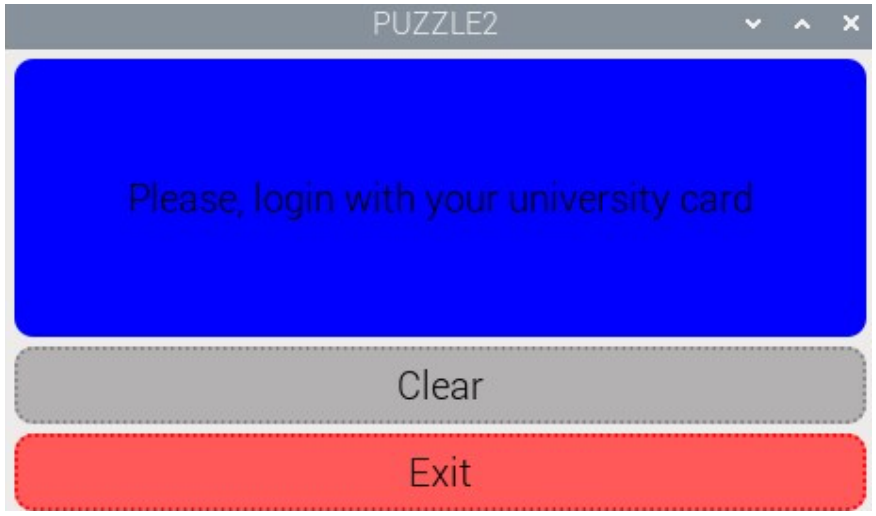
```
    def start_window(self):
        self.start_boxes()
        self.start_labels()
        self.start_buttons()
        self.wM.configure_style_CSS()
        self.start_reading_thread()
        self.show_all()

    def update_window(self,uid):
        self.wM.change_background_color(self.welcome_label,GREEN_COLOR)
        self.welcome_label.set_text(f"uid: {uid}")

    def reset_window(self,widget):
         self.wM.change_background_color(self.welcome_label,BLUE_COLOR)
         self.welcome_label.set_text(WELCOME_STRING)
         self.myReader.uid = None
        self.start_reading_thread()

class widgetManager:
    def __init__(self):
        self.editor_css = Gtk.CssProvider()

    def create_label(self,text):
      return Gtk.Label(label=text)

    def create_box(self,orientation,spacing):
        return Gtk.Box(orientation=orientation,spacing=spacing)
```

```
def create_button(self,text):
   return Gtk.Button(label=text)

def add_widget_box_start(self,box,widget,expand,fill,padding):
    box.pack_start(widget, expand, fill, padding)

def add_widget_box_end(self,box,widget,expand,fill,padding):
    box.pack_end(widget, expand, fill, padding)

def set_widget_name(self,widget,name)
    widget.set_name(name)

def change_backgroud_color(self,widget,color)
    widget.override_background_color(Gtk.StateFlags.NORMAL,color)

def configure_style_CSS(self):
  css = b"""
    #welcome_label{
        background-color: blue;
        color: black;
        padding: 60px;
        border-radius: 10px;
        margin-left: 5px;
        margin-right: 5px;
        margin-top 5px;
        font-size: 20;
        }
    #exit_button{
        background-color: #FF5959;
        color: black;
```

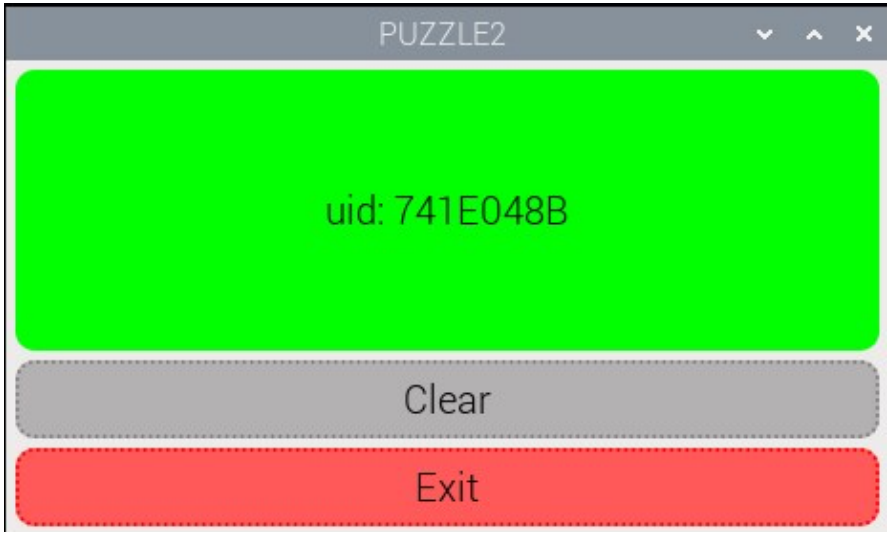
```
            padding: 5px;
            border-radius: 10px;
            border: 2px dotted red;
            margin-left: 5px;
            margin-right: 5px;
            margin-bottom: 5px;
            font-size: 20px;
            }
        #clear_button{
            background-color: #B4B1B2;
            color: black;
            padding: 5px;
            border-radius: 10px;
            border: 2px dotted gray;
            margin-left: 5px;
            margin-right: 5px;
            margin-bottom: 5px;
            margin-top: 5px;
            font-size: 20px;
            }
         """
        self.editor_css.load_from_data(css)
        screen = Gdk.Screen.get_default()
Gtk.StyleContext.add_provider_for_screen(screen,self.editor_css,Gtk.STYLE_PROVIDER_PRIORITY_APPLICATION)
```

```
class Application(Gtk.Application):
    def __init__(self):
        super().__init__()

    def do_activate(self):
        self.window = MyWindow(widgetManager())
        self.window.configure_window(400,100,Gtk.WindowPosition.CENTER,"PUZZLE2")
        self.window.connect("destroy", Gtk.main_quit)
        self.window.start_window()
        self.window.present()
        Gtk.main()

if __name__ == "__main__":
       app = Application()
       app.run()
```

La interfície configurada es veu de la següent manera:



Interfície esperant a que s'apropi un carnet UPC.



Interfície un cop s'apropa un carnet UPC.