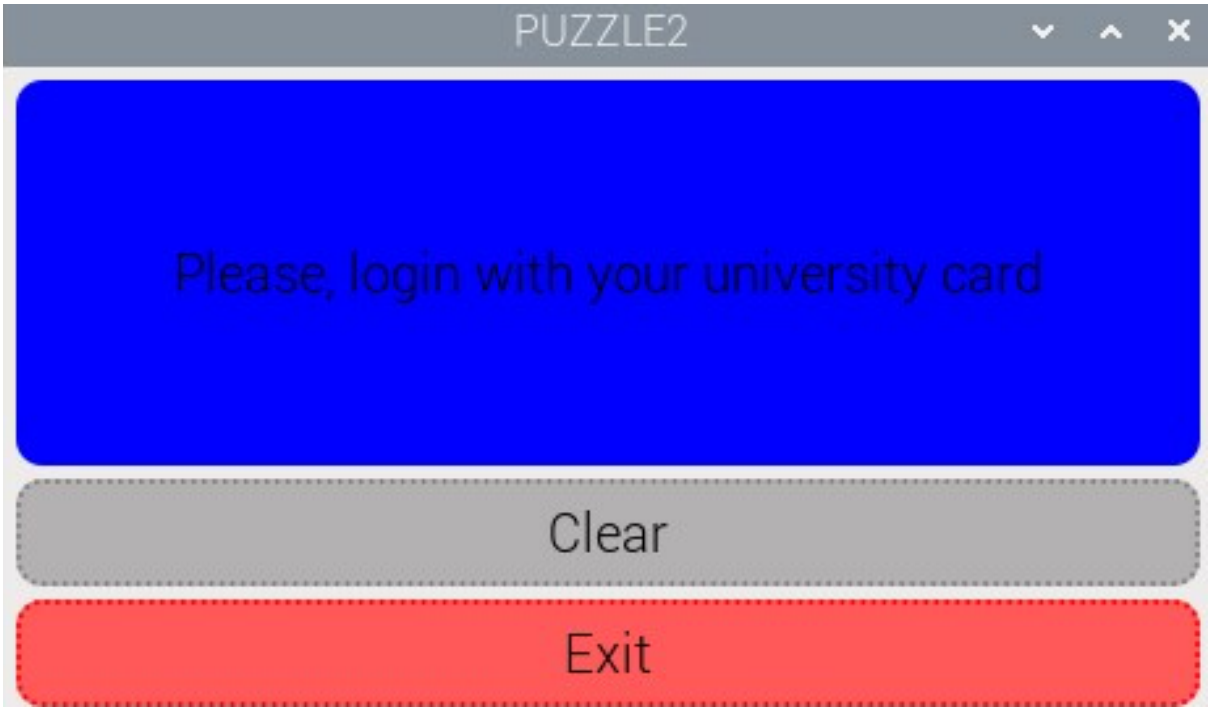
MEMÒRIA PUZZLE 2

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 i Gdk al nostre script 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** ens permet obtenir una referencia a la finestra de la aplicació per després aplicar-li les regles CSS.
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 pel nostre llenguatge de programació.
Tots els paquets instal·lats a diferència del puzzle 1 s'instal·len des de el gestor de paquets del sistema APT.

També es necessita importar el puzzle1 al puzzle2. A la meva carpeta inicialment es trobava tant el fitxer puzzle1.py com el puzzle2.py, per tant per importar un script al altre podía fer simplement
import puzzle1 al script del puzzle2. Aquest forma, tot i que funciona, depèn de que el puzzle1 es trobi al mateix directori que el script que executem. Per tant, vaig crear una biblioteca que inclou el fitxer *puzzle1.py* i pot ser importada per *puzzle2.py* sense necessitat de que estiguin al mateix directori.
Inicialment no sabia com fer-ho, creava els mòduls però al fer el import, el compilador no el trobava. Així que vaig pensar que podria crear la biblioteca a la carpeta on per defecte se'm va instalar el paquet *mfrc522* al puzzle anterior, ja que quan al puzzle1 feia:
from mfrc522 **import** SimpleMFRC522, el compilador **si** trobava la biblioteca.

Per tant, aquí creo un nou directori anometat “puzzle1Llibreria”, que serà el nom de la meva biblioteca.

```
hugo@raspberrypi:/usr/local/lib/python3.11/dist-packages $ ls
mfrc522  mfrc522-0.0.7.dist-info  puzzle1Llibreria  RPi  rpi_gpio-0.7.1.dist-info
```

Per tal que el directori es consideri com un mòdul, s’ha de crear un fitxer *__init__.py* a dintre (pot estar buit). Insereixo aquí el fitxer *puzzle1.py.*

```
hugo@raspberrypi:/usr/local/lib/python3.11/dist-packages/puzzle1Llibreria $ ls
__init__.py  puzzle1.py  __pycache__
hugo@raspberrypi:/usr/local/lib/python3.11/dist-packages/puzzle1Llibreria $
```

Ara ja des de el fitxer *puzzle2.py* puc importar la classe *Rfid_522()* i accedir a la funcionalitat programada al anterior puzzle.
Per importar aquesta classe de la nova biblioteca al fitxer puzzle2 he fet:

from puzzle1Llibreria.puzzle1 **import** Rfid_522

CODI

He volgut dividir el codi en dues classes, una per l'aplicació, que hereda els mètodes de la classe *Gtk.Application,* que activa la finestra i la propia finestra que hereda els mètodes de la classe *Gtk.ApplicationWindow.*

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…
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. Fa 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 i les regles d’estil CSS es troben a continuació.

Python

FITXER estils.css

El #<nom> coincideix amb el ID amb el que s'identifiquen els diferentes widgets, per tal que les regles s'apliquin de forma independent a cada widget.

```
       #welcome_label{
             background-color: blue;
             color: black;
             padding: 60px;
             border-radius: 10px;
             margin-left: 5px;
             margin-right: 5px;
             margin-top: 5px;
             font-size: 20px;
      }
      #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;
      }
      #accepted_label{
            background-color: green;     #Modifiquem el fons del label
             color: black;
             padding: 60px;
             border-radius: 10px;
             margin-left: 5px;
             margin-right: 5px;
             margin-top: 5px;
             font-size: 20px;
       }
```

Python

```
"""
En el puzzle2 es demana crear una versió gràfica del puzzle1. S'utilitzarà la la biblioteca PyGObject. PyGObject es un
binding que permet utilitzar les llibreries basades en GObject com GTK3 que estàn escrites amb C però desde Python.
Per instalar aquesta biblioteca fem: sudo pip3 install PyGObject. Seguidament, descarreguem el paquet gtk3 que es el que
conté les funcions per crear la interfaç. sudo apt install python3-gi (bindings de GTK para python. Finalment, instalem el
paquet gir1.2-gtk-3.0. Aquest paquet permet a la llibreria PyGObject accedir a les funcionalitats de la llibreria gtk3
desde Python.
sudo apt install gir1.2-gtk-3.0
"""
import gi
import puzzle1
import threading
gi.require_version("Gtk", "3.0")
from puzzle1Llibreria.puzzle1 import Rfid_522
from gi.repository import Gtk, GLib, Gdk

WELCOME_STRING = "Please, login with your university card"

"""
Classe per configurar la finestra d'una aplicació i els seus elements. La classe hereda els mètodes de la classe
Gtk.ApplicationWindow
"""
class MyWindow(Gtk.ApplicationWindow):

    def __init__(self):
        super().__init__()
        self.myReader = Rfid_522()
        self.editor_css = Gtk.CssProvider()   #Aquest objecte permet gestionar i aplicar els estils CSS
```

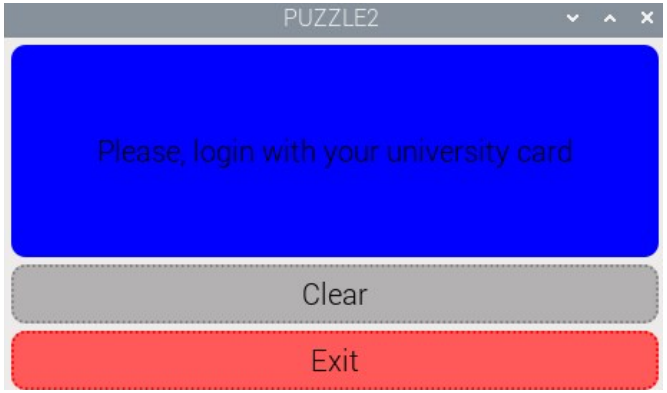
```
    """
    Configura la finestra amb els paràmetres escollits.
    Paràmetres:
        :amplada:    Amplada de la finestra
        :altura:     Altura de la finestra
        :posició: Posició de la finestra a la pantalla
        :titol: Títol de la finestra
    """

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

    """
    Creem la capsa principal y la afeguim a la finestra
    """
    def start_boxes(self):
        self.main_box = Gtk.Box(orientation=Gtk.Orientation.VERTICAL,spacing=0)
        self.add(self.main_box)

    """
    Instancia els labels que es faràn servir inicialment i els introdueix a la capsa principal
    """
    def start_labels(self):
        self.welcome_label = Gtk.Label(label=WELCOME_STRING)
        self.main_box.pack_start(self.welcome_label, False, False, 0)     #Afegim label a la part superior de la capsa
        self.welcome_label.set_name("welcome_label")                       #ID del label (per aplicar regles CSS)
```

```
"""
Instancia els botons que es faràn servil inicialmente. Els afegeix a la capsa
"""
def start_buttons(self):
    self.exit_button = Gtk.Button(label="Exit")
    self.exit_button.connect("clicked",Gtk.main_quit)
    self.main_box.pack_end(self.exit_button, False, False, 0)          #Afegim el label a la part inferior de la capsa
    self.exit_button.set_name("exit_button")

    self.clear_button = Gtk.Button(label="Clear")
    self.clear_button.connect("clicked",self.reset_window)
    self.main_box.pack_end(self.clear_button, False, False, 0)
    self.clear_button.set_name("clear_button")

"""
Crea i arrenca el fil auxiliar.
"""
def start_reading_thread(self):
    self.thread = threading.Thread(target=self.rf_reading_task)
    self.thread.daemon = True                                  #El Thread terminarà sempre quan es tanqui la finestra
    self.thread.start()

 """
   Funció que executarà el thread auxiliar. GTK no es thread-safe, per tant per evitar problemes hem de actualitzar la
   interfaç des de el fil principal, no desde el secundari.
"""
def rf_reading_task(self):
    self.myReader.read_uid()                                    #Llegim la uid
    GLib.idle_add(self.update_window, self.myReader.uid)       #Des del fil secundari executem la funció al principal
```

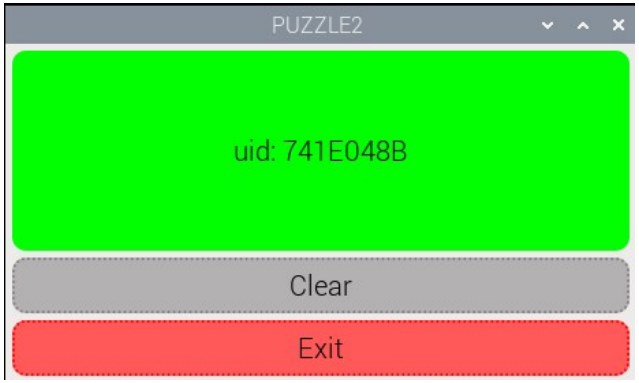
```
"""
   Iniciem tots els widgets i apliquem les regles CSS. Iniciem el thread auxiliar per llegir el carnet UPC i mostrem
   tots els widgets de la finestra.
"""
def start_window(self):
    self.start_boxes()
    self.start_labels()
    self.start_buttons()
    self.configure_style_CSS()
    self.start_reading_thread()            #Iniciem el thread auxiliar de lectura
    self.show_all()

"""
Un cop es detecta una lectura, es modifica el label de benvinguda i es mostra el uid per pantalla.
Paràmetres:
    :uid: Identificador de la tarjeta obtingut a la lectura.
"""
def update_window(self,uid):
    self.welcome_label.set_name("accepted_label")       #Modifiquem la ID del label per aplicar-li altres regles CSS
    self.welcome_label.set_text(f"uid: {uid}")

"""
Torna la finestra a l'estat inicial un cop polsem el botó "Clear".
"""
def reset_window(self,widget):
     self.welcome_label.set_name("welcome_label") #Tornem a la ID inicial per aplicar les regles CSS inicials
     self.welcome_label.set_text(WELCOME_STRING)
     self.myReader.uid = None              #Esborrem la uid anterior
     self.start_reading_thread()           #Tornem a iniciar el thread de lectura
```

```python
    """
Funció que aplica les regles CSS als widgets. El mètode set_widget_name realitzat sobre el label i els botons ho he fet
per ara poder aplicar el selector #<widget> per tal d'aplicar regles CSS individuals a cada widget.
    """
    def configure_style_CSS(self):
       self.editor_css.load_from_path("estils.css")  #Carreguem regles CSS del fitxer "estils.css"
       screen = Gdk.Screen.get_default()             #Obtenim una referència a la pantalla de la aplicació.
       Gtk.StyleContext.add_provider_for_screen(screen,self.editor_css,Gtk.STYLE_PROVIDER_PRIORITY_APPLICATION)

"""""""""""""""""""""""""""""""""""""""""""""""""""
Classe que permet gestionar la aplicació principal.
"""""""""""""""""""""""""""""""""""""""""""""""""""
class Application(Gtk.Application):
    def __init__(self):
        super().__init__()
       """
       Aquest mètode s'executarà quan sobre un objecte de la classe Gtk.Application executem .run()
       """
    def do_activate(self):
        self.window = MyWindow()
        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 en l'estat inicial.



Interfície un cop s'apropa un carnet UPC.