Puzzle2

Objectius: El principal objectiu en aquesta part del projecte és crear una interfície gràfica per tal d'aplicar el puzle 1 i poder mostrar el uid d'una targeta qualsevol. Per dur a terme el propòsit principal haurem de disposar d'un controlador gràfic, en el meu cas he utilitzat VNC Viewer, i per realitzar el codi s'utilitzarà la llibreria PyGObject. A més a més, utilitzarem CSS per millorar l'estètica del nostre entorn gràfic tal com es mostra a la següent figura:

Llibreries: La llibreria principal d'aquest puzzle és la gtk3. Aquesta llibreria ens permet desenvolupar l'entorn gràfic. Per instal·lar-la a la raspberry utilitzo el següent comando a la terminal:

```
sudo apt install python3-gi python3-gi-cairo gir1.2-gtk-3.0
```

Implementació del codi:

Per aconseguir realitzar l'objectiu d'aquest puzzle, em vaig organitzar per diferents parts. La primera va ser crear la Window i aprendre el funcionament d'aquesta nova llibreria. La segona part va ser crear el codi per tal de ajuntar-ho amb el puzzle1 a través d'un thread. I per últim un cop ja em va funcionar el programa en si, millorar l'estètica a través de CSS.

```python
import gi
import threading

gi.require_version("Gtk", "3.0")
from gi.repository import Gtk
from gi.repository import Gdk


class MyWindow(Gtk.Window):

    def __init__(self):
        super().__init__(title="Puzzle2")
        self.box = Gtk.Box(orientation=Gtk.Orientation.VERTICAL, spacing=6)
        self.add(self.box)

        self.label1 = Gtk.Label(label="Please log in with your university card")
        self.label1.set_size_request(800,200)
        self.label1.set_name("label1")
        self.box.pack_start(self.label1, True, True, 0)

        self.button2 = Gtk.Button(label="Clear")
        self.button2.connect("clicked", self.clear)

        self.box.pack_start(self.button2, True, True, 0)

win = MyWindow()
win.connect("destroy", Gtk.main_quit)
win.show_all()
Gtk.main()
```

Aquest és el codi que vaig implementar per fer la primera part. Com es pot observar les primeres línies de codi corresponen a la importació de les llibreries adients per poder dur a terme el codi. Tot seguit faig una classe MyWindow en la que creo un Box i modifico la orientació per tal de que sigui vertical. Després poso una label i un botó el qual serà per fer el clear. Amb això aconsegueixo dividir al box en dues parts.

Un cop he realitzat la classe, creo un objecte tipus MyWindow. Per mostrar la finestra per pantalla utilitzo "win.show_all" i per tal que s'elimini en el moment que cliquem la creu utilitzo `win.connect("destroy", Gtk.main_quit)`. Finalment, escric GTk.main per tal que faci el programa.

La segona part com he dit anteriorment consta de realitzar el programa que connecta el puzzle2 amb el nostre entorn gràfic. Vaig realitzar el següent codi:

```
class MyWindow(Gtk.Window):

    def __init__(self):
        super().__init__(title="Puzzle2")
        #Creació de la window
        self.box = Gtk.Box(orientation=Gtk.Orientation.VERTICAL, spacing=6)
        self.add(self.box)
        self.label1 = Gtk.Label(label="Please log in with your university card")
        self.label1.set_size_request(800,200)
        #Afegeixo la etiqueta principal
        self.label1.set_name("label1")
        self.box.pack_start(self.label1, True, True, 0)
        #Afegeixo el boto de clear
        self.button2 = Gtk.Button(label="Clear")
        self.button2.connect("clicked", self.clear)
        self.box.pack_start(self.button2, True, True, 0)


        #creació del primer thread
        self.thread=threading.Thread(target=self.uid, daemon=True)
        self.thread_in_use=True
        self.thread.start()
    #cua de threads
    def cuathread(self):
        GLib.idle_add(self.uid)

    #crea un thread nou i l'afegeix a la cua
    def createthread(self):
        thread= threading.Thread(target=self.cuathread)
        thread.start()

    #funcio per tornar a la pantalla inicial i tornar a començar el programa
    def clear(self, widget):
        self.label1.set_text("Please log in with your university card")
        self.thread=threading.Thread(target=self.createthread,daemon=True)
        self.thread.start()

    #funció per fer la funció del puzzle1
    def uid(self):
        pn532=RFID()
        card=pn532.read_uid()
        self.label1.set_text("UID:  "+card.upper())
```

Per començar és important importar el puzzle1 com si fos una llibreria. Per començar creo la funció `def uid(self):` en la que afegeixo la funció del puzzle1 i canvio el text de la label1. Tot seguit creo la funció de clear en la que primer de tot es canviarà el text de la etiqueta al text inicial i dsp crearà un nou thread amb el target= self.cuathread la qual posarà al thread nou a la cua utilitzant `GLib.idle_add(self.uid)`.

Per últim, vaig utilitzar CSS per estilitzar l'entorn gràfic i vaig afegir les següents línies:

```
self.blue = b"""

#label1{
    background-color: #29c8f6;
    font: bold 24px Calibri;
    color:#FFFFFF;
    }

"""
self.red = b"""

 #label1{
    background-color: #dd1432;
    font: bold 24px Calibri;
    color: #FFFFFF;
    }



"""

self.css_provider = Gtk.CssProvider() #Adding styles
self.css_provider.load_from_data(self.blue)
self.context = Gtk.StyleContext()
self.screen = Gdk.Screen.get_default()
self.context.add_provider_for_screen(self.screen, self.css_provider, Gtk.STYLE_PROVIDER_PRIORITY_APPLICATION)
```

El que faig és crear dues funcions CSS en les que canvio l'estil de la label1. Canviarà el background-color, la font i el color de la lletra. Després, per poder realitzar aquests canvis creo un objecte tipus Gtk.CssProvider() i utilitzo la funció "load_from_data(self.blue)" per començar amb la window de color blau. A més a més, inicialitzo un context de tipus Gtk.StyleContext() i una screen de tipus Gtk.Screen.() per tal de poder fer servir la funció "add_provider_for_screen". Un cop ja ha estat inicialitzat tot, cada vegada que es vol canviar de color s'ha de cridar la funció següent: "self.css_provider.load_from_data(self.red)" en el cas que vulguem posar la label1 de color vermell i "self.css_provider.load_from_data(self.blue)" en el cas que vulguem posar la label1 de color blau.

Problemes:

L'únic problema que he tingut en aquest puzzle2 2 ha estat que al principi quan premia el botó clear abans que es llegís una targeta em saltava un error i el programa es bloquejava. És per això que he creat la variable thread_in_use. Aquesta variable s'activa en el moment que inicia la lectura i es posa en False en el moment que ja s'ha imprès el uid per pantalla. La funció clear només retornarà a la pàgina inicial si aquesta variable esta en False.

Codi complert:

```
from puzzle1 import RFID
import gi
import threading

gi.require_version("Gtk", "3.0")
from gi.repository import Gtk
from gi.repository import Gdk
from gi.repository import GLib


class MyWindow(Gtk.Window):

    def __init__(self):
        super().__init__(title="Puzzle2")
        self.box = Gtk.Box(orientation=Gtk.Orientation.VERTICAL, spacing=6)
        self.add(self.box)

        self.label1 = Gtk.Label(label="Please log in with your university card")
        self.label1.set_size_request(800,200)
        self.label1.set_name("label1")
        self.box.pack_start(self.label1, True, True, 0)

        self.button2 = Gtk.Button(label="Clear")
        self.button2.connect("clicked", self.clear)

        self.box.pack_start(self.button2, True, True, 0)

        #define blue style
        self.blue = b"""

        #label1{
            background-color: #29c8f6;
            font: bold 24px Calibri;
            color:#FFFFFF;
            }

        """
        self.red = b"""

         #label1{
            background-color: #dd1432;
            font: bold 24px Calibri;
            color: #FFFFFF;
            }

        """

        self.css_provider = Gtk.CssProvider() #Adding styles
        self.css_provider.load_from_data(self.blue)
        self.context = Gtk.StyleContext()
        self.screen = Gdk.Screen.get_default()
        self.context.add_provider_for_screen(self.screen, self.css_provider, Gtk.STYLE_PROVIDER_PRIORITY_APPLICATION)
```

```
        self.css_provider = Gtk.CssProvider() #Adding styles
        self.css_provider.load_from_data(self.blue)
        self.context = Gtk.StyleContext()
        self.screen = Gdk.Screen.get_default()
        self.context.add_provider_for_screen(self.screen, self.css_provider, Gtk.STYLE_PROVIDER_PRIORITY_APPLICATION)


        self.thread=threading.Thread(target=self.uid, daemon=True)
        self.thread_in_use=True
        self.thread.start()

    def cuathreads(self):
        GLib.idle_add(self.uid)


    def clear(self, widget):
        if(self.thread_in_use==False):
            self.label1.set_text("Please log in with your university card")
            self.css_provider.load_from_data(self.blue)
            self.thread=threading.Thread(target=self.cuathreads, daemon=True)
            self.thread.start()
            self.thread_in_use=True


    def uid(self):
        pn532=RFID()
        card=pn532.read_uid()
        self.label1.set_text("UID:  "+card.upper())
        self.thread_in_use=False
        self.css_provider.load_from_data(self.red)

win = MyWindow()
win.connect("destroy", Gtk.main_quit)
win.show_all()
Gtk.main()
```