Anna Llamas Royo

PUZZLE 2

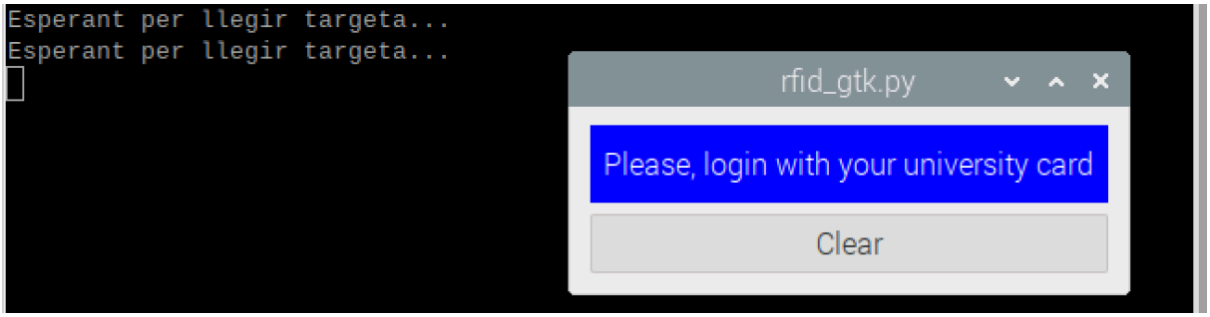
Python

```python
from PUZZLE1_ANNA import RfidPyNFC

import gi
gi.require_version('Gtk', '3.0')
import time
import threading
from gi.repository import Gtk, GLib, Gdk
import binascii
import struct
import pynfc

gi.require_version('Gtk', '3.0')

class NFCApp(Gtk.Window):
    def _init_(self):
        Gtk.Window._init_(self, title="rfid_gtk.py")
        self.set_border_width(10)
        self.set_default_size(300, 100)

        # Crear el Label
        self.label = Gtk.Label(label="Please, login with your university
card")

        self.label.override_background_color(Gtk.StateFlags.NORMAL,
Gdk.RGBA(0, 0, 1, 1))  # Fons blau
        self.label.override_color(Gtk.StateFlags.NORMAL, Gdk.RGBA(1, 1, 1,
1))  # Text en blanc

        # Crear botó Clear
        self.button = Gtk.Button(label="Clear")
        self.button.connect("clicked", self.clear_label)

        # Crear layout en forma vertical
        vbox = Gtk.Box(orientation=Gtk.Orientation.VERTICAL, spacing=6)
        vbox.pack_start(self.label, True, True, 0)
        vbox.pack_start(self.button, False, False, 0)
        self.add(vbox)

        # Inicialitzar el lector NFC i llegir el UID
        self.nfc_reader = RfidPyNFC()
        self.start_nfc_thread()

    def start_nfc_thread(self):
        self.thread = threading.Thread(target=self.check_nfc)
        self.thread.daemon = True
        self.thread.start()
```

```
    def check_nfc(self):
        while True:
            uid = self.nfc_reader.read_uid()
            if uid:
                GLib.idle_add(self.update_label, uid)

    def update_label(self, uid):
        self.label.set_text(f"uid: {uid}")
        self.label.override_background_color(Gtk.StateFlags.NORMAL,
Gdk.RGBA(1, 0, 0, 1))  # Fons vermell
        self.label.override_color(Gtk.StateFlags.NORMAL, Gdk.RGBA(1, 1, 1,
1))  # Text en blanc

    def clear_label(self, widget):
        self.label.set_text("Please, login with your university card")
        self.label.override_background_color(Gtk.StateFlags.NORMAL,
Gdk.RGBA(0, 0, 1, 1))  # Fons blau
        self.label.override_color(Gtk.StateFlags.NORMAL, Gdk.RGBA(1, 1, 1,
1))  # Text en blanc

# Crear i executar
win = NFCApp()
win.connect("destroy", Gtk.main_quit)
win.show_all()
Gtk.main()
```

Aquest és el Label que apareix quan executem el codi, on podem veure que ens demana que apropem la targeta de la universitat.

Una vegada apropem la targeta, ens mostra el UID de la mateixa.



Quan apretem el botó de Clear es reinicia el Label i ens torna a demanar que apropem la targeta de la universitat.

Per provar una altra targeta, ara he apropat la targeta que ve amb el component.

Codi comentat

Primerament faig els imports necessaris per fer un entorn gràfic (GUI) fent servir GTK i la llibreria **threading** que fa que la lectura NFC s'executi en un fil separat evitant que es congeli l'entorn gràfic.

Dins del mètode **read_uid**, he fet servir la funció self.nfc.poll() per detectar un objecte NFC.

Per configurar la finestra principal de l'entorn gràfic, he fet servir la funció **Gtk.Window._init_** i **self.label** per crear l'etiqueta que surt en el requadre i canviar el color del fons, a blau. Després he fet servir la funció **self.button** pel botó Clear i després **vbox** per estructurar el Label i el botó en una columna vertical. Després **self.start_nfc_thread** per tenir contínuament el lector "activat" sense bloquejar l'entorn gràfic.

He hagut de crear el mètode **start_nfc_thread** perquè se'm congelava l'entorn gràfic i d'aquesta manera, es crea un fil que treballa en segon pla, on el lector espera la targeta. He fet servir el mètode **check_nfc** que he creat per estar contínuament detectant una targeta i llegint el UID.

Finalment cal el mètode d'actualitzar el Label, imprimeix el UID i canvia el color del fons a vermell.