Sílvia Ruano Ferrer
PBE – Puzzle 2

Memòria Puzzle 2

1. Objectiu

Aquesta part del projecte consisteix en una aplicació gràfica desenvolupada en Ruby amb la llibreria GTK3. La funcionalitat principal és la creació d'una interfície per a un sistema de registre mitjançant targetes d'identificació, que demana a l'usuari que passi la seva targeta RFID per llegir el seu identificador únic (UID). Aquesta aplicació té diverses finestres, una on es demana la targeta i una altra que es mostra el UID de la targeta. Per fer possible aquesta funcionalitat, he separat el codi en diferents mètodes i utilitzat fils d'execució per gestionar la lectura del UID sense bloquejar la interfície gràfica.

2. Instal·lació de les Dependències

Abans de començar a programar, cal instal·lar la llibreria GTK3, que proporciona les eines per construir la interfície gràfica de l'usuari (GUI). Per instal·lar-la, simplement s'ha d'executar la següent comanda: sudo gem install gtk3 . La resta de llibreries i paquets necessaris ja els tenia instal·lats prèviament en la implementació del puzzle 1.

3. Estructura del Codi i Funcionalitats

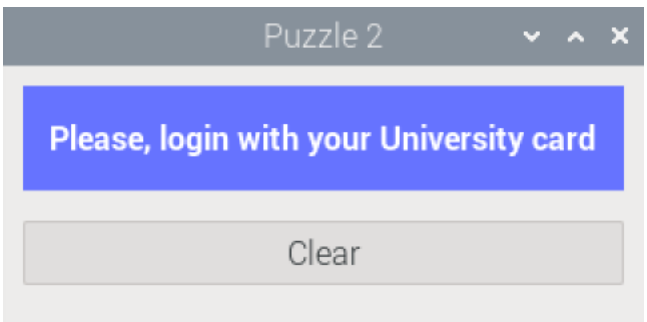
Per mantenir el codi més organitzat he separat les diferents accions en mètodes dins de la classe Finestres.

3.1. Create_Window

Aquest mètode és la base per crear qualsevol finestra de l'aplicació. Posant-li un títol, un text descriptiu i un color de fons personalitzat, que dependran de cada finestra que es creï. També afegeix un botó "Clear" que, quan es clica, tanca la finestra actual i obre la finestra de login, que es defineix al mètode fin1. Això permet que l'usuari pugui tornar fàcilment a la primera finestra per iniciar de nou el procés de registre.

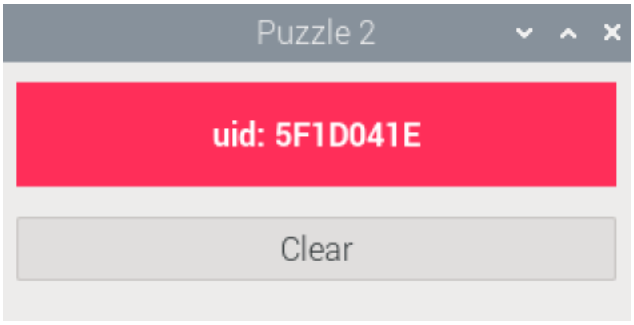
3.2. Fin1

Aquest mètode és la que mostra la finestra inicial, on es demana a l'usuari que passi la seva targeta RFID per registrar-se. Dins, es crida a la funció read_uid en un thread separat per llegir el UID de la targeta de manera que no es bloquegi el fil principal d'execució, el que controla la GUI.



3.3. Fin2

Aquest mètode crea una nova finestra on es mostra el UID llegit de la targeta, una vegada ha obtingut el valor a través del lector. Aquesta finestra just es mostra després de fer el procés de lectura amb el lector de NFC.



3.4. Funció del Botó "Clear"

El botó "Clear", que es crea dins del mètode create_window, permet que l'usuari torni a la finestra inicial en qualsevol moment. Quan es clica, el botó tanca la finestra actual i crida fin1 perquè l'usuari pugui tornar a veure el missatge de login i iniciar el procés de lectura amb una nova targeta.

Després de clicar el botó, la lectura de la següent targeta no es pot fer gaire de pressa, i s'han de deixar uns segons de marge, perquè sinó es bloqueja i tanca totes les finestres en lloc de llegir la nova targeta.

3.5. Lectura del UID amb Threads

Un punt important d'aquesta aplicació és el mètode read_uid, definit prèviament en el puzle1. Aquest mètode permet comunicar-se amb el lector RFID i llegir el UID de la targeta, però és una operació bloquejant, que podria fer que la interfície gràfica es quedés penjada mentre es fa la lectura. Per evitar-ho, he implementat un fil d'execució auxiliar que s'encarrega d'executar read_uid sense afectar el fil principal.

A fin1, s'inicia un fil auxiliar amb Thread-new per executar el mètode del puzzle1. Així, la lectura del UID es fa en segon pla sense interferir amb la GUI. Quan el fil obté el UID, s'ha de passar aquest valor al fil principal per mostrar-lo a la finestra de la GUI. Però per evitar bloquejar el fil he utilitzat GLib::Idle.add , que permet passar el resultat sense interrompre la GUI. Aquest mètode assegura que la GUI només s'actualitzi des del fil principal, evitant errors i problemes de sincronització entre threads. Quan GLib::Idle.add rep el resultat, fin2 s'encarrega de crear la finestra que mostra el UID

3.6. Codi

```
require "gtk3"
require_relative 'puzzle1'

class Finestres
        def create_window(title, label_text, background_color)
                window=Gtk::Window.new(title)
                window.set_border_width(10)

                vbox = Gtk::VBox.new(:vertical, 5)

                label = Gtk::Label.new
                label.set_markup("<b>#{label_text}</b>")
                label.override_background_color(:normal, Gdk::RGBA.new(*background_color))
                label.override_color(:normal, Gdk::RGBA.new(1,1,1,1))
                label.set_size_request(300, 50)
                vbox.pack_start(label,expand: true,fill: true , padding: 0);

                button = Gtk::Button.new(:label =>  "Clear")
                button.signal_connect("clicked") do
                        window.destroy #cerrar ventana actual
                        fin1 #obrir finestra de login
                end
                vbox.pack_start(button, expand: false, fill: false, padding: 10);

                window.add(vbox)

                window.signal_connect("delete-event") { |_widget| Gtk.main_quit }
                window.show_all
                return window #per poderla tancar despres
        end

        def fin1
                window1 = create_window("Puzzle 2" , "Please, login with your University card" , [0.4,0.45,1,1])
                lector = Rfid.new
                Thread.new do #crear un nou thread per executar el codi bloquejant
                        uid = lector.read_uid
                        GLib::Idle.add do #torna al thread grfic per fer actualitzacions de la GUI
                                window1.destroy
                                fin2(uid)
                                false #evitar que la GLib::Idle es repeteixi
                        end
                end
        end

        def fin2(uid)
                create_window("Puzzle 2" , "uid: #{uid}",[1,0.18,0.35,1])
        end
end

if __FILE__ == $0
  Finestres.new.fin1

  Gtk.main

end
```