# PUZZLE 2

## Codi:

Com que el codi es una mica llarg l'explicaré pas per pas:

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

# Permite cerrar con Crtl+C
import signal
signal.signal(signal.SIGINT, signal.SIG_DFL)
```

*import gi*: Importa el sistema de enllaços PyGObject per utilitzar GTK a Python.

*gi.require_version("Gtk", "3.0")*: Assegura que s'utilitzi GTK 3.

*from gi.repository import Gtk, Gdk*: Importa els móduls necessaris de GTK.

*signal.signal(signal.SIGINT, signal.SIG_DFL):* Habilita que es pugui tancar l' app amb Ctrl + C desde la terminal.

```
class UIDReaderApp(Gtk.Window):
    def __init__(self):
        super().__init__(title="puzzle_2.py")
        self.set_default_size(400, 200)
```

Defineix una finestra GTK personalitzada que l'anomenem UIDReaderApp en aquest cas, després crida al constructor de Gtk.Window y li posa el nom de puzzle_2.py.

```
self.default_text = "Please, login with your university card"

box = Gtk.Box(orientation=Gtk.Orientation.VERTICAL, spacing=10, margin=20)
self.add(box)
```

*self.default_text:* Text que es mostrarà mentres no es llegeixi cap targeta.

*Gtk.Box(...):* crea un contenedor vertical amb separació (spacing=10) y marges de 20 píxels.

*self.add(box):* Afegeix aquest contenedor a la finestra principal.

```python
# Área de mensaje
self.message_box = Gtk.EventBox()
self.label = Gtk.Label(label=self.default_text)
self.message_box.add(self.label)
box.pack_start(self.message_box, True, True, 0)

# Campo oculto para leer el UID
self.entry = Gtk.Entry()
self.entry.set_visibility(False)
self.entry.connect("activate", self.on_uid_entered)
self.entry.connect("realize", self.on_entry_ready)
box.pack_start(self.entry, False, False, 0)

# Botón Clear
clear_btn = Gtk.Button(label="Clear")
clear_btn.connect("clicked", self.reset)
box.pack_start(clear_btn, False, False, 0)

# Botón Exit
quit_btn = Gtk.Button(label="Exit")
quit_btn.connect("clicked", Gtk.main_quit)
box.pack_start(quit_btn, False, False, 0)

# Estilos
self.apply_css()
self.set_style("blue")
```

**# Area de Mensaje**

Aquí utilitzo *Gtk.EventBox* per poder aplicar fons de color (ja que GTK no permet posar color directament a Gtk.Label). Després s'afegeix el text que pertoca ( UID o missatge inicial) dins de la zona amb el fons de color i s'afegeix al layout.

**#Campo oculto lectura UID**

Primer crea un camp de text, i després oculta el que s'escriu en aquest mateix camp.

Amb *connect("activate", self.on_uid_entered)*: quan es presiona Enter, es crida al métode que processa l'UID. Després s'espera a que estigui completament carregat y ho amagem y enfoquem. Finalment, afegeix el camp de text al layout encara que no es vegi per a que funcioni.

**#Boton Clear**

Crea un botó **"Clear"**, que quan es clica crida al métode self.reset que esborra l'UID, torna a posar el fons a blau amb el missatge d'inici, i enfoca el camp ocult per poder tornar a llegir una altra targeta

**#Boton Exit**

Crea un botó **"Exit"**, que quan es clica es tanca la app cridant a Gtk.main_quit().

(En teoria no fa falta aquest botó, ja que es pot sortir amb Crtl+C pero crec que queda bé y no es molt complicat d'implementar)

**#Estilos**

Crida al métode que aplica estils CSS personalitzats (vermell y blau). Y comença amb el blau.

```
def on_uid_entered(self, entry):
    uid = entry.get_text().strip()
    uid_int = int(uid)
    uid_hex = hex(uid_int)[2:].upper()
    self.label.set_text(f"uid: {uid_hex}")
    self.set_style("red")
    entry.set_text("")
```

Aquesta seria la part del puzzle 1, no he utilitzat el puzzle com a llibreria perque l'he hagut de modificar. Al puzzle 1 utilitzaba la funció *getpass.getpass()* que és bloquejant.

He decidit canviar això per el codi de *on_uid_entered* que no és bloquejant, ja que s'executa automàticament quan el lector termina d'escriure, i fa que el codi no estigui esperant activament sino que simplement reaccioni quan ocurreix la lectura.

Quan el lector tecleja l'UID y fa Enter, Gtk genera un event *activate* i es respon amb *on_uid_entered*. (Com es pot veure a la foto anterior a #Campo oculto para leer el UID)

D'aquesta manera no calen usar threads auxiliars i s'eviten problemes de concurrència.

```
def reset(self, _):
    self.label.set_text(self.default_text)
    self.set_style("blue")
    self.entry.grab_focus()
```

Torna a mostrar el text inicual, fica el fons de color blau i enfoca el camp ocult per la següent lectura.

```
def on_entry_ready(self, widget):
    widget.hide()
    widget.grab_focus()
```

Es crida automàticament quan el camp està completament inicializtat.

L'oculta visualment (encara que segueixi actiu).

```
def set_style(self, color):
    ctx = self.message_box.get_style_context()
    ctx.remove_class("blue")
    ctx.remove_class("red")
    ctx.add_class(color)
```

Cambia el color del fons (blau o vermell) de  l'àrea de missatge.

```
def apply_css(self):
    css = b"""
    .blue {
        background-color: #0000cc;
        color: white;
        font-size: 20px;
        padding: 20px;
    }
    .red {
        background-color: #cc0000;
        color: white;
        font-size: 20px;
        padding: 20px;
    }
    """
    style = Gtk.CssProvider()
    style.load_from_data(css)
    Gtk.StyleContext.add_provider_for_screen(
        Gdk.Screen.get_default(), style, Gtk.STYLE_PROVIDER_PRIORITY_APPLICATION
    )
```

Defineix els estils CSS per a .blue i .red, i aplica els estils per a tota la pantalla GTK.

```
if __name__ == "__main__":
    app = UIDReaderApp()
    app.connect("destroy", Gtk.main_quit)
    app.show_all()
    Gtk.main()
```

Crea la app, conecta l'esdeveniment de tancar la finestra per sortir del programa.

Mostra tot (show_all()), i inicia el bucle principal de GTK (Gtk.main()).

## Resultat: