

Universidad Autónoma de Tamaulipas

Facultad de Ingeniería Tampico



ASIGNATURA

PROGRAMACION DE INTERFACES Y

PUERTOS

6to. Semestre – Grupo “i”
2025 -1

TRABAJO

Desarrollo de Prácticas y Proyectos

UNIDAD

2 -DESARROLLO E IMPLEMENTACIÓN DE CONTROLADORES MEDIANTE
INTERFACES Y PUERTOS

Docente: Dr. García Ruiz Alejandro H.

Integrante del Equipo	Nivel de Participación
Izaguirre Cortes Emanuel	33.33%
Turrubiates Mejia Gilberto	33.33%
García Salas Yahir Misael	33.33%
Total:	100%

Índice

Índice	1
Repositorio(s) de Prácticas	2
PP1.	2
Descripción de la practica	2
Introducción	2
Componentes para el desarrollo de la practica	2
Desarrollo	3
Pruebas Realizadas	3
Conclusión PP1.....	5
PP2.	6
Descripción de la practica	6
Introducción	6
Componentes para el desarrollo de la practica	6
Desarrollo	6
Pruebas Realizadas	7
Conclusión PP2.....	8

Repositorio(s) de Prácticas

Practica	Repositorio
Practica 1 a la 2	https://github.com/Emanuel-Izaguirre-Cortes-03/Pip_2025_1_eq_4_i/tree/main/Practicas

PP1.

Esta práctica consiste en el desarrollo de una aplicación de escritorio en Python utilizando PyQt5. La aplicación permite mostrar diferentes imágenes según la selección de un usuario mediante un QSlider y verificar si el texto ingresado por el usuario coincide con la imagen mostrada.

Descripción de la practica

Esta práctica consiste en el desarrollo de una aplicación de escritorio en Python utilizando PyQt5. La aplicación permite mostrar diferentes imágenes según la selección de un usuario mediante un QSlider y verificar si el texto ingresado por el usuario coincide con la imagen mostrada.

Introducción

Las interfaces gráficas de usuario (GUI) facilitan la interacción con aplicaciones mediante elementos visuales. PyQt5 es un framework popular para la creación de interfaces en Python. En esta práctica, se desarrolló una aplicación donde el usuario debe identificar correctamente el nombre de una imagen seleccionada aleatoriamente.

Componentes para el desarrollo de la practica

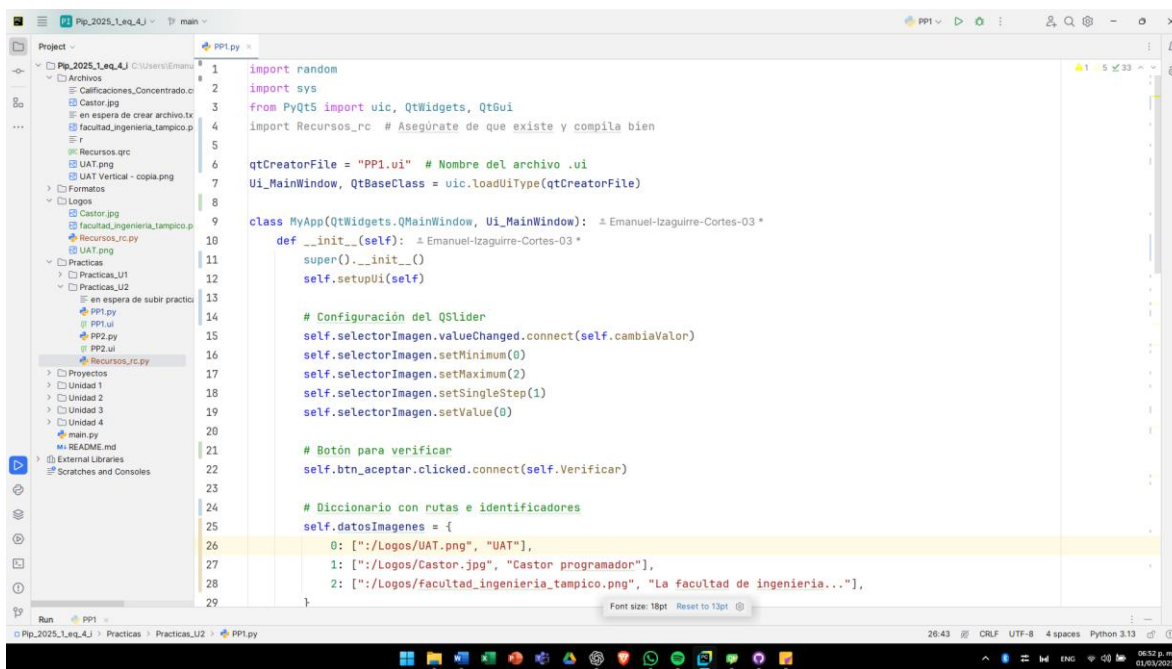
- **Lenguaje de programación:** Python 3
- **Bibliotecas:** PyQt5, random, sys
- **Widgets de PyQt5:** QLabel, QSlider, QPushButton, QMessageBox
- **Archivos:** Archivo .ui para la interfaz diseñada en Qt Designer y un script .py para la lógica de la aplicación

Desarrollo

1. Se diseñó la interfaz en Qt Designer y se guardó en un archivo .ui.
2. Se cargó la interfaz en Python usando `uic.loadUiType()`.
3. Se implementó un diccionario con las rutas de las imágenes y sus nombres.
4. Se configuró un QSlider para cambiar las imágenes según su valor.
5. Se implementó un método para seleccionar aleatoriamente una imagen y mostrar su nombre en un QLineEdit.
6. Se programó la verificación de la coincidencia entre el nombre ingresado y el correcto.

Pruebas Realizadas

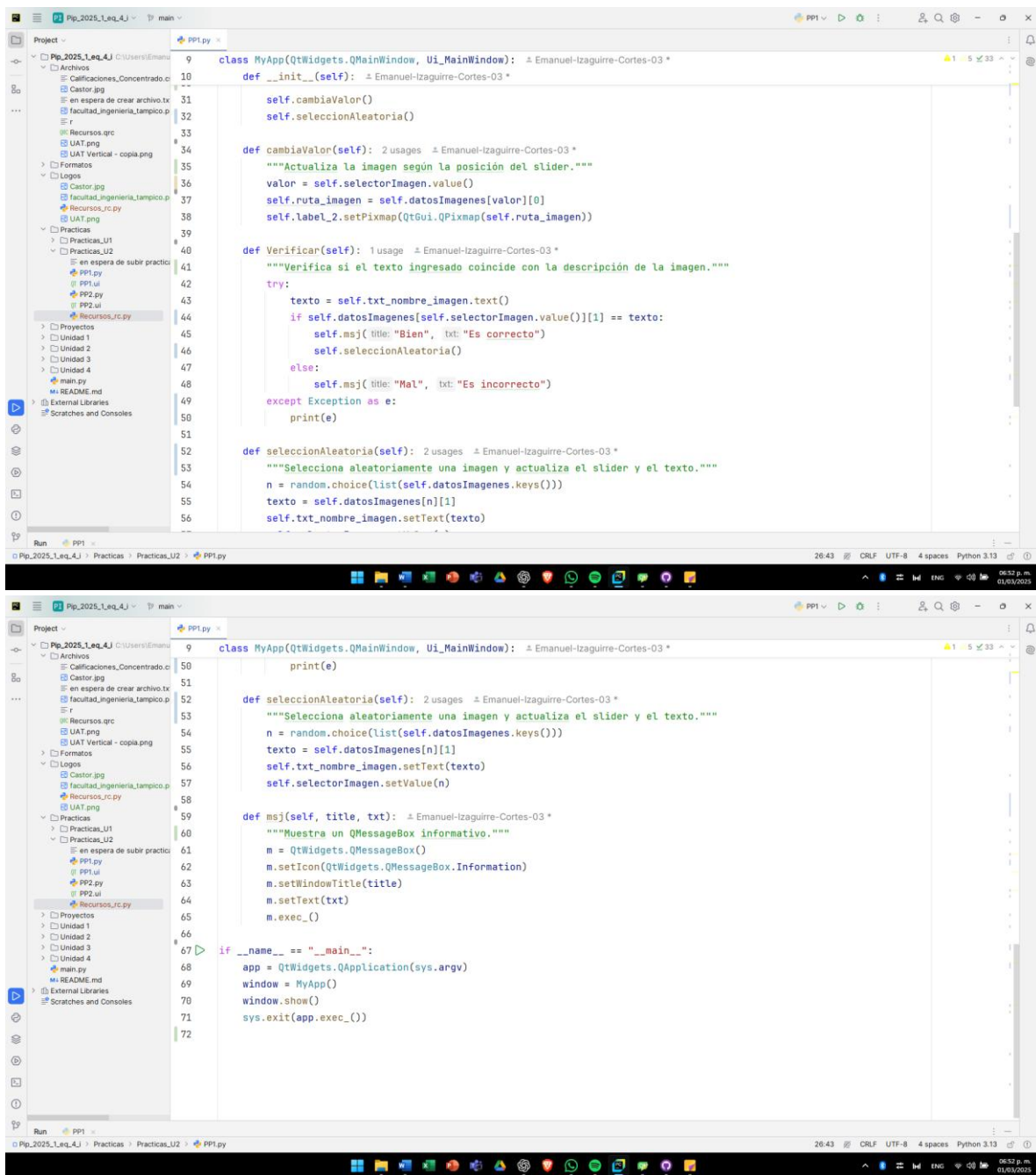
- Se probó la carga de imágenes según la selección del slider.
- Se verificó la correcta selección aleatoria de imágenes.
- Se comprobó la validación de nombres de imágenes con mensajes informativos.
- Se corrigieron errores en la comparación del nombre ingresado.



```

1 import random
2 import sys
3 from PyQt5 import uic, QtWidgets, QtGui
4 import Recursos_rc # Asegúrate de que existe y compila bien
5
6 qtCreatorFile = "PPI.ui" # Nombre del archivo .ui
7 Ui_MainWindow, QtBaseClass = uic.loadUiType(qtCreatorFile)
8
9 class MyApp(QtWidgets.QMainWindow, Ui_MainWindow): # Emanuel-Izaguirre-Cortes-03 *
10     def __init__(self): # Emanuel-Izaguirre-Cortes-03 *
11         super().__init__()
12         self.setupUi(self)
13
14         # Configuración del QSlider
15         self.selectorImagen.valueChanged.connect(self.cambiaValor)
16         self.selectorImagen.setMinimum(0)
17         self.selectorImagen.setMaximum(2)
18         self.selectorImagen.setSingleStep(1)
19         self.selectorImagen.setValue(0)
20
21         # Botón para verificar
22         self.btn_aceptar.clicked.connect(self.Verificar)
23
24         # Diccionario con rutas e identificadores
25         self.datosImagenes = {
26             0: ["/Logos/UAT.png", "UAT"],
27             1: ["/Logos/Castor.jpg", "Castor programador"],
28             2: ["/Logos/facultad_ingenieria_tampico.png", "La facultad de ingeniería..."],
29         }

```



```

class MyApp(QMainWindow, Ui_MainWindow):
    def __init__(self):
        self.cambiaValor()
        self.seleccionAleatoria()

    def cambiaValor(self):
        """Actualiza la imagen según la posición del slider."""
        valor = self.selectorImagen.value()
        self.ruta_imagen = self.datosImagenes[valor][0]
        self.label_2.setPixmap(QtGui.QPixmap(self.ruta_imagen))

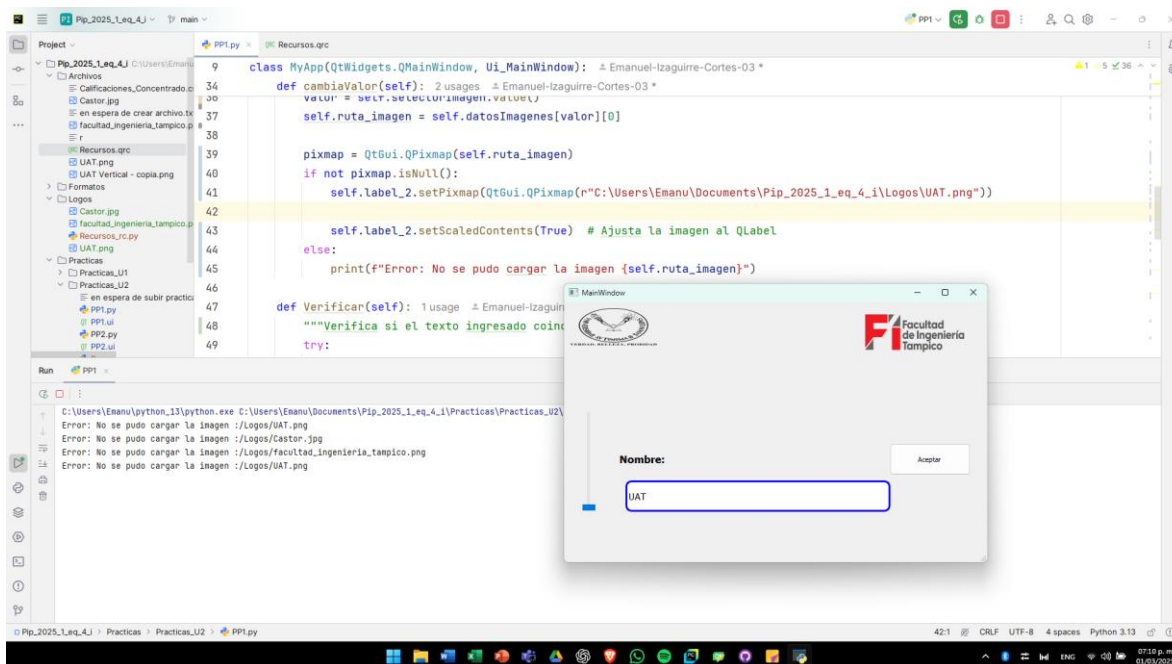
    def Verificar(self):
        """Verifica si el texto ingresado coincide con la descripción de la imagen."""
        try:
            texto = self.txt_nombre_imagen.text()
            if self.datosImagenes[self.selectorImagen.value()][1] == texto:
                self.msj(title="Bien", txt="Es correcto")
                self.seleccionAleatoria()
            else:
                self.msj(title="Mal", txt="Es incorrecto")
        except Exception as e:
            print(e)

    def seleccionAleatoria(self):
        """Selecciona aleatoriamente una imagen y actualiza el slider y el texto."""
        n = random.choice(list(self.datosImagenes.keys()))
        texto = self.datosImagenes[n][1]
        self.txt_nombre_imagen.setText(texto)

    def msj(self, title, txt):
        """Muestra un QMessageBox informativo."""
        m = QtWidgets.QMessageBox()
        m.setIcon(QtWidgets.QMessageBox.Information)
        m.setWindowTitle(title)
        m.setText(txt)
        m.exec_()

if __name__ == "__main__":
    app = QtWidgets.QApplication(sys.argv)
    window = MyApp()
    window.show()
    sys.exit(app.exec_())

```



Conclusión PP1

En esta práctica, se implementó una interfaz gráfica utilizando **PyQt5**, permitiendo la manipulación de imágenes a través de un **QSlider** y verificando la correspondencia entre el nombre ingresado y la imagen mostrada.

Al concluir el desarrollo, se identificaron varios aspectos clave:

1. **Manejo de Archivos UI:** La correcta carga del archivo .ui con `uic.loadUiType()` facilita la integración de diseños creados en Qt Designer, reduciendo el tiempo de desarrollo.
2. **Gestión de Rutas de Imágenes:** Se evidenció la importancia de definir rutas de imágenes correctamente, evitando errores relacionados con secuencias de escape (`\U`) en Windows.
3. **Interactividad con el Usuario:** La validación de texto y el uso de `QMessageBox` para mostrar mensajes de acierto o error mejoraron la experiencia del usuario, permitiendo una retroalimentación inmediata.
4. **Aleatorización de Datos:** Se utilizó la función `random.choice()` para seleccionar imágenes de manera aleatoria, lo que diversificó la interacción con el programa y reforzó conceptos de estructuras de datos como diccionarios en Python.
5. **Importancia del Debugging:** Se resolvieron errores de sintaxis y se optimizó la gestión de eventos, resaltando la necesidad de pruebas constantes para garantizar un funcionamiento correcto.

En general, la práctica permitió reforzar conocimientos en **interfaces gráficas, manipulación de imágenes, eventos en GUI y validación de datos**, habilidades esenciales para el desarrollo de software interactivo.

PP2.

Descripción de la practica

La aplicación permite iniciar un temporizador que se actualiza cada segundo y muestra el tiempo transcurrido en una etiqueta. Se ha desarrollado utilizando PyQt5 y un diseño de interfaz definido en un archivo de código Python.

Introducción

Este proyecto es una aplicación en Python con una interfaz gráfica (GUI) desarrollada con PyQt5. La aplicación implementa un temporizador que muestra el tiempo transcurrido en segundos.

Componentes para el desarrollo de la practica

Lenguaje de programación**: Python

- **Bibliotecas**:

- PyQt5 (para la interfaz gráfica)
- QtCore (para el temporizador)
- QtWidgets (para los elementos de la interfaz)

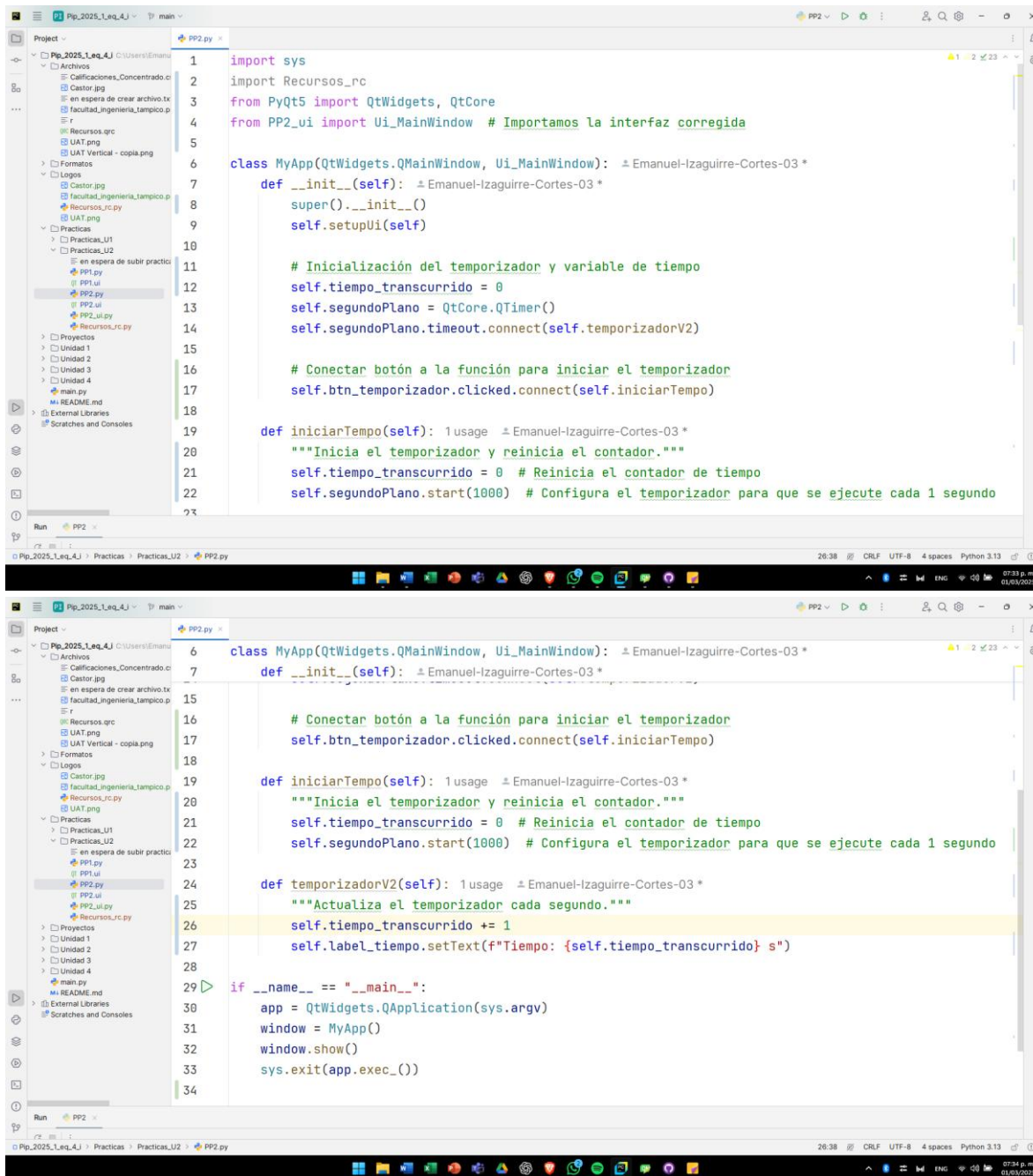
- **Archivos**:

- `PP2_ui.py`: Define la interfaz gráfica.
- `PP2.py`: Contiene la lógica del programa y la interacción con la interfaz.

Desarrollo

1. Se creó la interfaz gráfica en `PP2_ui.py`, definiendo un botón (`btn_temporizador`) y una etiqueta (`label_tiempo`).
2. Se implementó la lógica en `PP2.py`, donde:
 - Se inicializa la interfaz y los componentes.
 - Se configura un temporizador que se actualiza cada segundo.
 - Se maneja el evento del botón para iniciar el temporizador.

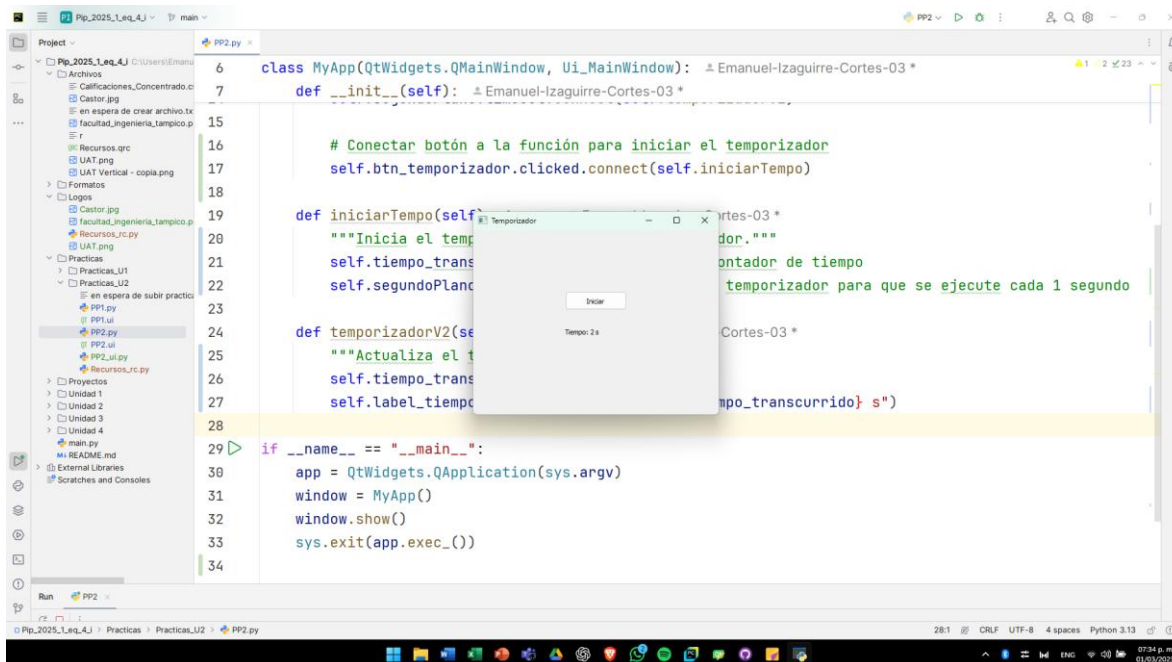
Pruebas Realizadas



```

1 import sys
2 import Recursos_rc
3 from PyQt5 import QtWidgets, QtCore
4 from PP2_ui import Ui_MainWindow # Importamos la interfaz corregida
5
6 class MyApp(QtWidgets.QMainWindow, Ui_MainWindow): # Emanuel-Izaguirre-Cortes-03 *
7     def __init__(self): # Emanuel-Izaguirre-Cortes-03 *
8         super().__init__()
9         self.setupUi(self)
10
11         # Inicialización del temporizador y variable de tiempo
12         self.tiempo_transcurrido = 0
13         self.segundoPlano = QtCore.QTimer()
14         self.segundoPlano.timeout.connect(self.temporizadorV2)
15
16         # Conectar botón a la función para iniciar el temporizador
17         self.btn_temporizador.clicked.connect(self.iniciarTempo)
18
19     def iniciarTempo(self): 1 usage # Emanuel-Izaguirre-Cortes-03 *
20         """Inicia el temporizador y reinicia el contador."""
21         self.tiempo_transcurrido = 0 # Reinicia el contador de tiempo
22         self.segundoPlano.start(1000) # Configura el temporizador para que se ejecute cada 1 segundo
23
24 class MyApp(QtWidgets.QMainWindow, Ui_MainWindow): # Emanuel-Izaguirre-Cortes-03 *
25     def __init__(self): # Emanuel-Izaguirre-Cortes-03 *
26
27         # Conectar botón a la función para iniciar el temporizador
28         self.btn_temporizador.clicked.connect(self.iniciarTempo)
29
30     def iniciarTempo(self): 1 usage # Emanuel-Izaguirre-Cortes-03 *
31         """Inicia el temporizador y reinicia el contador."""
32         self.tiempo_transcurrido = 0 # Reinicia el contador de tiempo
33         self.segundoPlano.start(1000) # Configura el temporizador para que se ejecute cada 1 segundo
34
35     def temporizadorV2(self): 1 usage # Emanuel-Izaguirre-Cortes-03 *
36         """Actualiza el temporizador cada segundo."""
37         self.tiempo_transcurrido += 1
38         self.label_tiempo.setText(f"Tiempo: {self.tiempo_transcurrido} s")
39
40 if __name__ == "__main__":
41     app = QtWidgets.QApplication(sys.argv)
42     window = MyApp()
43     window.show()
44     sys.exit(app.exec_())

```

```

6 class MyApp(QMainWindow, Ui_MainWindow):
7     def __init__(self):
15
16     # Conectar botón a la función para iniciar el temporizador
17     self.btn_temporizador.clicked.connect(self.iniciarTempo)
18
19     def iniciarTempo(self):
20         """Inicia el temporizador"""
21         self.tiempo_transcurrido = 0
22         self.segundoPlanificado = 1
23         self.tiempo_transcurrido = 0
24         self.tiempo_transcurrido = 0
25         self.tiempo_transcurrido = 0
26         self.tiempo_transcurrido = 0
27         self.tiempo_transcurrido = 0
28
29 if __name__ == "__main__":
30     app = QApplication(sys.argv)
31     window = MyApp()
32     window.show()
33     sys.exit(app.exec_())
34

```

Conclusión PP2

El proyecto cumple con su objetivo de implementar un temporizador simple con una interfaz gráfica en PyQt5. Se logró una integración funcional entre la lógica del programa y la interfaz, permitiendo futuras mejoras o expansión.

