

**PORTAFOLIO DE PRÁCTICA: Practica 7 -  
FileDialog, QPushButton,  
QListWidget, QTabWidget y QLabel  
Programación Visual**

**FACILITADOR:  
ALI PÉREZ GÓMEZ**

**SEMESTRE: CUARTO SEMESTRE**

**INTEGRANTES:  
Lopez Fuentevilla Suheidy  
López Melchor Itzel Yared  
Prieto Vargas Luis Eduardo**

## Índice

Introducción.....	pag 3
Objetivo .....	pag 4
Marco teorico.....	pag 4
Simulación .....	pag 9
Recolección de información.....	pag 9
Análisis de resultado.....	pag 9
Conclusiones.....	pag10
Referencia.....	pag10

## Introducción

El siguiente documento tiene como fin evidenciar el reporte de las evidencias realizadas en las prácticas de Programación Visual dando una breve explicación de la practica realizada en el aula.

Como una introducción breve de lo que se podrá encontrar en este documento de evidencias será un poco del procedimiento del código en Python y su diseño en QT designer.

## Objetivo

Este ejemplo muestra cómo usar botones, listWidget, TabWidget y label para seleccionar carpetas y archivos para obtener su ruta, necesaria para procesarlos. La listWidget es una lista que permite seleccionar los elementos en ella. El TabWidget permite organizar varias ventanas en diferentes pestañas y los label son etiquetas.

## Marco teórico

**QPushButton** es un componente que permite a los usuarios interactuar con la aplicación mediante clics. Es uno de los widgets más básicos y comunes en una interfaz gráfica. Cuando el usuario presiona el botón, este emite señales que pueden ser conectadas a funciones o métodos para realizar acciones específicas.

### Características Principales

Texto e Íconos:

Texto: Puedes asignar un texto al botón que se muestra en su superficie.

Ícono: Puedes agregar un ícono al botón junto con el texto o solo un ícono.

Señales y Slots:

clicked(): Emitida cuando el botón es presionado y liberado.

pressed(): Emitida cuando el botón es presionado.

released(): Emitida cuando el botón es liberado.

toggled(bool checked): Emitida cuando un botón "checkable" cambia de estado.

Propiedades de Estilo:

Hojas de Estilo: Permite personalizar la apariencia del botón utilizando hojas de estilo (QSS), similar a CSS en web.

Iconos y Fuentes: Personaliza el ícono y el tipo de fuente.

Estados del Botón:

Habilitado/Deshabilitado: Controla si el botón puede ser interactuado.

Checkable: Permite que el botón tenga un estado alterno (presionado/no presionado).

**QLabel** es un widget de visualización que puede mostrar texto simple o enriquecido (usando HTML), así como imágenes. Este widget es útil para mostrar

etiquetas, encabezados, mensajes, imágenes o cualquier contenido visual que no necesite interacción del usuario.

#### Características Principales

##### Texto e Imágenes:

Texto: Puede mostrar texto simple o texto formateado utilizando HTML.

Imágenes: Puede mostrar imágenes, ajustarlas y escalarlas según sea necesario.

##### Interacción Limitada:

Aunque QLabel está diseñado principalmente para mostrar información, puede detectar algunos eventos, como el paso del ratón sobre él, y actuar como un enlace cuando se habilita.

##### Autoajuste de Tamaño:

Se puede configurar para ajustar automáticamente su tamaño al contenido que muestra, ya sea texto o una imagen.

##### Estilos:

Se puede personalizar su apariencia usando hojas de estilo (QSS).

**QListWidget** es un widget que muestra una lista de elementos, cada uno representado por un objeto QListWidgetItem. Es ideal para aplicaciones donde se requiere una lista de elementos que el usuario puede seleccionar, modificar o reorganizar.

#### Características Principales

##### Elementos de Lista:

Permite agregar, eliminar y manipular elementos de lista.

Cada elemento puede contener texto, íconos y datos personalizados.

##### Selección de Elementos:

Soporta selección simple, múltiple y de rango.

Permite controlar la selección mediante diversos modos de selección.

##### Ordenación y Reorganización:

Permite ordenar los elementos de la lista.

Soporta arrastrar y soltar para reorganizar elementos.

##### Sencillez de Uso:

Combina QListView y QStandardItemModel, proporcionando una API simplificada para la manipulación de listas.

**QTabWidget** permite crear una interfaz con múltiples páginas, cada una representada por una pestaña. Cada pestaña puede contener cualquier tipo de widget, como formularios, tablas, gráficos, etc. Esto es especialmente útil para aplicaciones que necesitan organizar gran cantidad de contenido en un espacio limitado.

## Características Principales

### Navegación por Pestañas:

Cada pestaña representa una página de contenido diferente.

Los usuarios pueden cambiar entre pestañas haciendo clic en ellas.

### Adición y Eliminación de Pestañas:

Las pestañas pueden añadirse y eliminarse dinámicamente.

Se pueden añadir íconos y textos descriptivos a cada pestaña.

### Posición de las Pestañas:

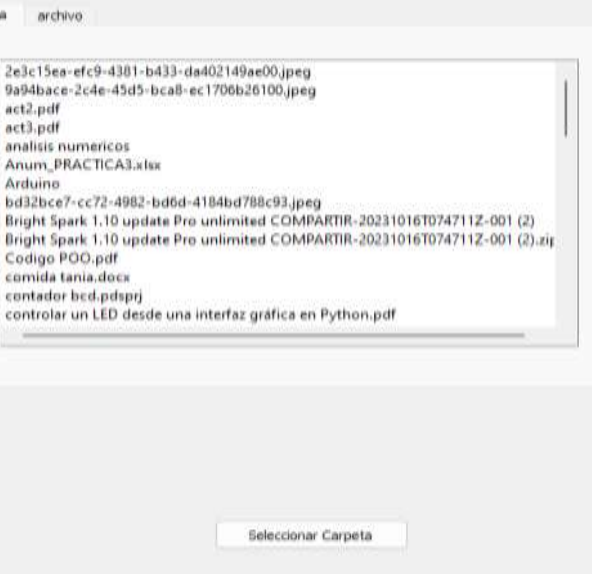
Las pestañas pueden estar ubicadas en la parte superior, inferior, izquierda o derecha del widget.

### Control de Interfaz:

Métodos y señales para controlar y reaccionar a cambios en las pestañas.

## Simulación

## Software

[illegible]

MainWindow

carpeta | archivo

- 2e3c15ea-efc9-4381-b433-da402149ae00.jpeg
- 9a94bace-2c4e-45d5-bca8-ec1706b26100.jpeg
- act2.pdf
- act3.pdf
- analisis numericos
- Anum\_PRACTICA3.xlsx
- Arduino
- bd32bce7-cc72-4982-bd6d-4184bd788c93.jpeg
- Bright Spark 1.10 update Pro unlimited COMPARTIR-20231016T074711Z-001 (2)
- Bright Spark 1.10 update Pro unlimited COMPARTIR-20231016T074711Z-001 (2).zip
- Codigo POO.pdf
- comida tania.docx
- contador bed.pdspj
- controlar un LED desde una interfaz gráfica en Python.pdf

Seleccionar Carpeta

## Recolección de información

La paractica realizada tenia como finalidad hacer un backend y un frontend y vincularlos usando QT designer y Python con visual studio code, ademas la función de este es la función carpeta permite seleccionar una carpeta para listar los archivos que contiene en la listWidget que se encuentra en la pestaña carpeta. Por otro lado, la función archivo permite seleccionar un archivo para mostrar el nombre del archivo seleccionado en el label que se encuentra en la pestaña archivo. Finalmente, la función mostrar\_archivo muestra el nombre del archivo seleccionado de la listWidget en el label que se encuentra en la pestaña carpeta.

## Análisis de resultado

La dificultad de la practica no es muchas sin embargo hay que tener en cuenta el nombre de cada widget que se agregue ya que en la parte del código debe llamarse igual, otra cosa que mencionar es que se debe tener el archivo generado desde el software de QT designer en la misma ruta donde se va a crear nuestro código en Python, de no ser así no se podrá importar el archivo.

## Conclusiones

Al final el objetivo de la practica se cumplió, otra cosa que añadir es que se debe contar con las librerías y los archivos en sus rutas necesarias para compilar el código, de lo contrario por más que el código este bien no te reconocerá las funciones.

## Referencias

Summerfield, M. (2008). Rapid GUI Programming with Python and Qt: The Definitive Guide to PyQt Programming. Prentice Hall.

Harvey, M. (2018). Qt5 Python GUI Programming Cookbook: Building responsive and powerful cross-platform applications with PyQt. Packt Publishing.

Dawson, M. (2019). PyQt6 By Example: Create modern GUIs with Python and Qt6. Packt Publishing.