

PROGRAMACIÓN CON PYTHON

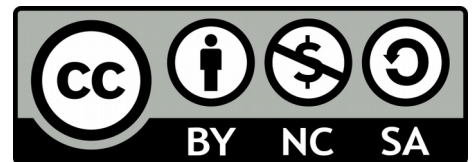
(CEFIRE CTEM)



Interfaces gráficas con Tkinter
Ejercicios voluntarios

Esta obra está sujeta a la licencia Reconocimiento-NoComercial-CompartirIgual 4.0 Internacional de Creative Commons. Para ver una copia de esta licencia, visitad

<http://creativecommons.org/licenses/by-nc-sa/4.0/>.



Autora: María Paz Segura Valero (mpazprofe@gmail.com)

CONTENIDO

1. Introducción.....	2
2. Ejercicios voluntarios.....	2
2.1. Ejercicio 1: Interfaz gráfica.....	3
2.2. Ejercicio 2: Lógica de la aplicación.....	4

1. Introducción

En este documento puedes encontrar una serie de **ejercicios voluntarios** para que pongas en práctica lo aprendido durante esta unidad.

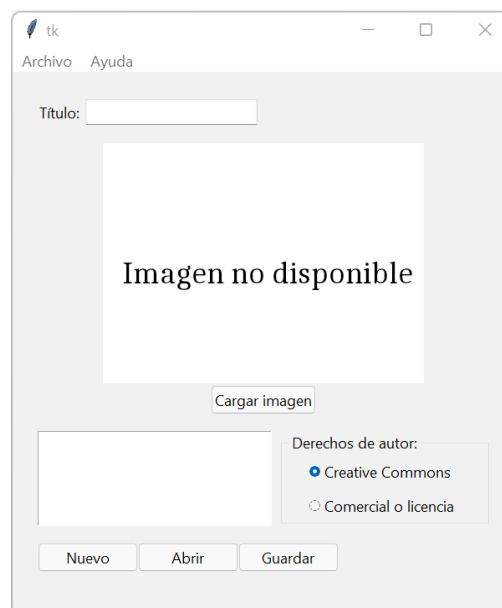
En el aula virtual dispondrás de las soluciones pero te recomiendo que intentes solucionarlos por ti mismo/a porque, aunque no sean obligatorios para superar el curso, sí pueden ayudarte a enfrentarte al *ejercicio obligatorio* del final.

Puedes utilizar el foro del curso para consultar tus dudas.

2. Ejercicios voluntarios

Durante estos ejercicios vamos a crear una aplicación que permitirá mostrar una imagen, su título, descripción y el tipo de licencia de la misma. Los datos de cada imagen estarán cargados en ficheros independientes, de tal forma que, el programa nos permitirá cargar los datos de un fichero, modificar su información y guardarlo. También podremos crear ficheros nuevos de imágenes.

El aspecto de la aplicación será el siguiente:



Vamos a dividir la faena en dos partes:

- En el primer ejercicio nos centraremos en crear los widgets que se mostrarán en la interfaz gráfica.
- En el segundo ejercicio le añadiremos la lógica/funcionalidad a la aplicación.

2.1. Ejercicio 1: Interfaz gráfica

Vamos a crear los *widgets* de la interfaz gráfica sin programar ningún comportamiento de la aplicación. Es decir, solo mostrará la parte gráfica pero no programaremos nada más.

Puedes crear el programa utilizando una **clase Aplicacion**, como en el ejemplo de la teoría, o sin ella. Tú decides.

Las características de los widgets se recogen en la siguiente tabla:

Widget	Características
Raíz	Ancho de 600 píxeles, altura de 650 píxeles y posicionado (X, Y) donde más te guste. Deberá tener 20 píxeles de margen entre el borde y su contenido.
Marcos de distribución	Crear cuatro <i>Frame</i> para poder distribuir el resto de widgets. Cada marco tendrá estas características: <ul style="list-style-type: none"> • Margen de 10 píxeles entre su borde y su contenido. • Rellenar (fill) todo el eje X. • Alineación (side) <i>top</i>.
Título	Etiqueta que muestre el texto «Título»: alineado a la izquierda. Campo de entrada corta que tenga asociada una variable de control de tipo <code>StringVar()</code> . Se mostrará en el primer marco.
Imagen	Etiqueta que muestre, por defecto, la imagen del fichero "imagen_no_disponible.png". Se mostrará en el segundo marco.
Botón Cargar	Botón que mostrará el texto «Cargar imagen» y ejecutará la función <code>cargar()</code> . Se mostrará en el segundo marco.
Descripción	Campo de entrada multilínea con un ancho de 23 caracteres y altura de 5. Se mostrará en el tercer marco alineado a la izquierda.
Derechos de autor	Marco de tipo <i>LabelFrame</i> que muestre el texto «Derechos de autor:». Se mostrará en el tercer marco alineado a la derecha. Dos <i>widgets</i> de tipo <i>Radiobutton</i> que mostrarán las opciones: <ul style="list-style-type: none"> • «Creative Commons» con valor 1 (valor por defecto) • «Comercial o licencia» con valor 2. Deberás asociarles una variable de control de tipo <code>IntVar()</code> .

	Se mostrarán en el LabelFrame anterior.
Botones inferiores	<p>Crear un botón que mostrará el texto «Nuevo» y ejecutará la función <code>nuevo()</code>.</p> <p>Crear un botón que mostrará el texto «Abrir» y ejecutará la función <code>abrir()</code>.</p> <p>Crear un botón que mostrará el texto «Guardar» y ejecutará la función <code>guardar()</code>.</p> <p>Se mostrarán en el cuarto marco alineados a la izquierda.</p>
Menú	<p>Crear el menú principal de la aplicación con las siguientes opciones:</p> <ul style="list-style-type: none"> Submenú <i>Archivo</i>: opciones <i>Nuevo</i>, <i>Abrir</i> y <i>Guardar</i>, un separador, opción <i>Salir</i>. <ul style="list-style-type: none"> Las opciones <i>Nuevo</i>, <i>Abrir</i> y <i>Guardar</i> ejecutarán las mismas funciones que los botones homónimos. La opción <i>Salir</i> provocará el fin de la aplicación. Submenú <i>Ayuda</i>: opción <i>Acerca de...</i> <ul style="list-style-type: none"> Esta opción llamará a la función <code>acerca_de()</code>.

Puedes crear el programa desde cero o basarte en el esquema que tienes en el fichero **ud5_ejer1_interfaz (ESQUEMA).py** del aula virtual. En este caso, deberás cambiar las instrucciones *pass* que encuentres en el código.

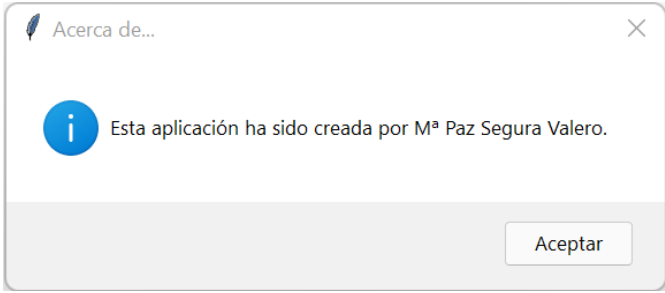
En el aula virtual dispones de la solución del ejercicio en el fichero **ud5_ejer1_interfaz.py**

2.2. Ejercicio 2: Lógica de la aplicación

Ahora que disponemos de la interfaz gráfica, podemos empezar a rellenar con código. En la siguiente tabla se describen las funciones.

Widget	Características
<code>reset()</code>	Esta función se encargará de reinicializar todos los widgets de la interfaz y será llamada desde otras funciones.
<code>cargar()</code>	<p>Mostrará un diálogo de tipo <code>AskOpenFileName</code> y permitirá seleccionar imágenes de tipo <code>.PNG</code>.</p> <p>Podemos utilizar como directorio inicial el directorio actual de la aplicación. Para ello podemos utilizar la función <code>getcwd()</code> de la librería os. Ejemplo: <code>dir_actual = os.getcwd()</code></p>

	La imagen seleccionada se mostrará en la etiqueta de imagen.
nuevo()	<p>Mostrará un diálogo <code>AskQuestion</code> con la siguiente pregunta: "Se borrarán los datos del formulario. ¿Estás seguro/a?"</p> <p>Si la respuesta es afirmativa entonces se llamará a la función <code>reset()</code>.</p>
abrir()	<p>Mostrará un diálogo <code>AskQuestion</code> con la siguiente pregunta: "Se borrarán los datos del formulario. ¿Estás seguro/a?"</p> <p>Si la respuesta es afirmativa entonces mostrará un diálogo de tipo <code>AskOpenFileName</code> y permitirá seleccionar ficheros de tipo <code>.TXT</code>.</p> <p>Podemos utilizar como directorio inicial el directorio actual de la aplicación.</p> <p>Si se ha seleccionado un fichero entonces se leerá su contenido para rellenar con él los widgets de la interfaz.</p> <p>El fichero contendrá una única línea con el siguiente formato: <code>título nombre_imagen descripción licencia</code></p> <p><u>Ejemplo:</u> <code>Polaroid polaroid.png Este dibujo vectorial... 1</code></p> <p>La línea tiene cuatro campos separados por el símbolo <code> </code>.</p> <p>Existe una función de cadenas de texto llamada <code>split()</code> que nos permite obtener una lista con todos los elementos de una cadena teniendo en cuenta un carácter de separación (blanco por defecto).</p> <p>Su <u>sintaxis</u> es la siguiente: <code>mi_cadena.split(carácter_separación)</code></p> <p><u>Ejemplo:</u> <pre>>>> linea = "Polaroid polaroid.png Este dibujo vectorial... 1" >>> linea.split(' ') ['Polaroid', 'polaroid.png', 'Este dibujo vectorial...', '1']</pre></p> <p>Puede que la línea del fichero acabe con el carácter de fin de línea <code>\n</code> y deberás tenerlo en cuenta.</p> <p>Si no se ha podido guardar en el fichero, se mostrará un diálogo de error.</p>

guardar()	<p>Mostrará un diálogo <code>AskSaveAsFilename</code> para preguntar el fichero que queremos actualizar con la información que haya en la interfaz. La extensión por defecto será <code>.txt</code>.</p> <p>Escribiremos una línea con el siguiente formato:</p> <pre>título nombre_imagen descripción licencia</pre> <p><u>Ejemplo:</u></p> <pre>Polaroid polaroid.png Este dibujo vectorial... 1</pre> <p>Recuerda que la línea tiene cuatro campos separados por el símbolo <code> </code>.</p> <p>Si no se ha podido guardar en el fichero, se mostrará un diálogo de error.</p> <p>Si se ha podido guardar, se mostrará un diálogo de información.</p>
acerca_de()	<p>Mostrar un diálogo de información con tus nombres y apellidos.</p> <p><u>Ejemplo:</u></p> 

Puedes crear el programa desde cero o basarte en el esquema que tienes en el fichero **ud5_ejer2_logica (ESQUEMA).py** del aula virtual. En este caso, deberás cambiar las instrucciones *pass* que encuentres en el código.

En el aula virtual dispones de la solución del ejercicio en el fichero **ud5_ejer2_logica.py**.