Alumno: Agustín Vallejo
Dni: 35.044.471

RecetasYa

¡no pidas delivery, cociná!

python™

UPATECO

Tecnicatura Universitaria en Desarrollo De Software

Upateco

ASÍ SE COCINÓ ESTA APP

¿Te gustaría tener todas tus recetas favoritas en un solo lugar y acceder a ellas fácilmente? Con nuestra aplicación de recetario en Tkinter, podrás hacer eso y mucho más. La interfaz gráfica intuitiva y fácil de usar te permitirá buscar y visualizar recetas en segundos, incluso puedes buscar por nombre de la receta. Con solo un clic, tendrás toda la información que necesitas para preparar deliciosas comidas.

Mi trabajo consiste en la creación de una aplicación de recetario utilizando la biblioteca Tkinter de Python. La aplicación cuenta con una interfaz gráfica que permite al usuario buscar y visualizar recetas.

En el código que le muestro, se definen tres clases: Receta, Ingrediente y App. La clase Receta contiene los atributos de una receta, como su nombre, ingredientes, tiempo de preparación, tiempo de cocción y fecha de creación. La clase Ingrediente define los atributos de un ingrediente, como su nombre, unidad de medida y cantidad. La clase App es la que se encarga de crear la interfaz gráfica y manejar las acciones del usuario.

```
from PIL import Image, ImageTk
import tkinter as tk
import csv
from tkinter import filedialog
import random
```

- **from PIL import Image, ImageTk:** Esta línea importa las clases Image e ImageTk de la biblioteca PIL, que se utiliza para trabajar con imágenes en Python.

- **import tkinter as tk:** Esta línea importa la biblioteca tkinter, que es una biblioteca de interfaz gráfica de usuario (GUI) estándar de Python. La as tk le da a la biblioteca un alias para que se pueda referenciar más fácilmente.

- **import csv:** Esta línea importa la biblioteca csv, que se utiliza para trabajar con archivos CSV (valores separados por comas) en Python.

- **from tkinter import filedialog:** Esta línea importa la función filedialog de tkinter. filedialog se utiliza para abrir cuadros de diálogo de selección de archivos en la GUI.

- **import random:** Esta línea importa el módulo random de Python, que se utiliza para generar aleatorios.

```python
class Receta:
    def __init__(self, nombre, ingredientes, preparacion, tiempo_preparacion, tiempo_coccion,fecha_
        self.nombre = nombre
        self.ingredientes = ingredientes
        self.preparacion = preparacion
        self.tiempo_preparacion = tiempo_preparacion
        self.tiempo_coccion = tiempo_coccion # Se agrega tiempo de cocción
        self.imagen = imagen
        self.fecha_creacion = fecha_creacion
    def __str__(self):
        return self.nombre
```

La clase "Receta" tiene un método constructor __init__ que inicializa los atributos de la instancia de la clase, que son:

nombre: el nombre de la receta
ingredientes: una lista de los ingredientes necesarios para la receta
preparacion: los pasos necesarios para preparar la receta
tiempo_preparacion: el tiempo que se tarda en preparar la receta
tiempo_coccion: el tiempo que se tarda en cocinar la receta
imagen: una imagen opcional de la receta
fecha_creacion: la fecha en que se creó la receta

El método __str__ simplemente devuelve el nombre de la receta como una cadena de caracteres.

```python
class Ingrediente:
    def __init__(self, nombre, unidad_medida, cantidad):
        self.nombre = nombre
        self.unidad_medida = unidad_medida
        self.cantidad = cantidad

    def __str__(self):
        return f"{self.nombre}, {self.cantidad}, {self.unidad_medida}"
```

La clase "Ingrediente" tiene un método constructor __init__ que inicializa los atributos de la instancia de la clase, que son:

nombre: el nombre del ingrediente
unidad_medida: la unidad de medida utilizada para la cantidad del ingrediente (por ejemplo, taza, gramo, etc.)
cantidad: la cantidad del ingrediente necesaria para la receta
El método __str__ devuelve una cadena de caracteres que representa el ingrediente en el formato "nombre, cantidad, unidad de medida". Por ejemplo, si el ingrediente es "harina" y se necesitan 2 tazas de ella, el método __str__ devuelve "harina, 2, tazas".

```python
class App:
    def __init__(self, root):
        self.root = root
        self.recetas = []
        # Carga la imagen y la establece como fondo de la ventana
        img = Image.open("images/brand4.png")
        photo = ImageTk.PhotoImage(img)
        label = tk.Label(self.root, image=photo)
        label.image = photo  # Esta línea previene que la imagen sea eliminada por el recolector de
        label.place(x=0, y=0, relwidth=1, relheight=1)
        self.imagen = Image.open("images/123.gif")
        self.frames = []
```

La clase "App" tiene un método constructor __init__ que inicializa los atributos de la instancia de la clase, que son:

root: la ventana principal de la aplicación
recetas: una lista vacía que se utilizará para almacenar objetos de la clase "Receta".
En este código también se realiza lo siguiente:

Carga y muestra una imagen de fondo en la ventana principal.
Carga y muestra una animación en la ventana principal.
Espera 3 segundos antes de mostrar la ventana principal.
Actualiza la animación en la ventana principal cada cierto tiempo.

```
try:
    while True:
        self.frames.append(ImageTk.PhotoImage(self.imagen))
        self.imagen.seek(len(self.frames))
except EOFError:
    pass
```

El bloque de código utiliza un bucle while para cargar cada uno de los frames de la animación en una lista llamada "frames". Esto se hace mediante el uso del método ImageTk.PhotoImage() de la biblioteca PIL (Python Imaging Library) que se utiliza para convertir cada frame de la animación en un objeto de imagen de Tkinter.

El método seek() se utiliza para desplazarse al siguiente frame en la animación. Si se llega al final de la animación, se lanza una excepción "EOFError" que se captura con un bloque except para evitar errores en la aplicación.

En resumen, este bloque de código carga y muestra una animación en la ventana principal de la aplicación mediante el uso de un bucle while y el método ImageTk.PhotoImage() de la biblioteca PIL..

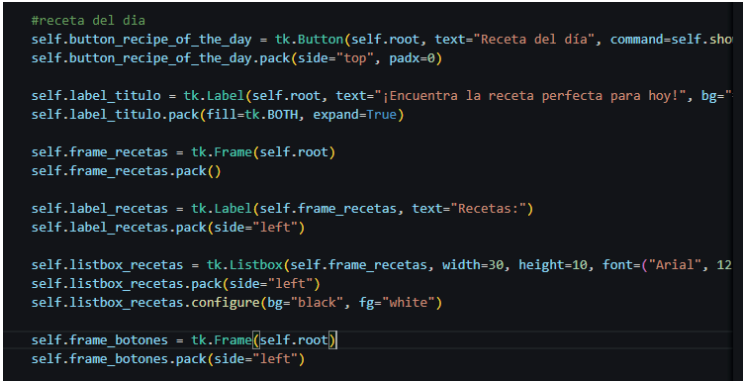
```
#busqueda
self.frame_busqueda = tk.Frame(self.root)
self.frame_busqueda.pack(side="top")

self.entry_busqueda = tk.Entry(self.frame_busqueda, width=35, bg="#747a8a", fg="#FFFFFF", bd
self.entry_busqueda.pack(side="left")

self.entry_busqueda.insert(0, "Buscar receta...")

self.button_buscar = tk.Button(self.frame_busqueda, text="Buscar", command=self.buscar_recet
                               font=('Arial Rounded MT Bold',8), padx=8, pady=5, cu
self.button_buscar.pack(side="left")
```

En este código se está creando una sección de búsqueda en la interfaz gráfica de usuario de la aplicación. Esto se hace utilizando el módulo "tkinter" de Python para crear widgets de la GUI.

Primero se crea un "Frame" que contendrá los elementos de la búsqueda. Se crea un cuadro de entrada ("Entry") para que el usuario pueda ingresar el término de búsqueda. Se establece un texto predeterminado en el cuadro de entrada utilizando el método "insert" del objeto "Entry".

Luego se crea un botón ("Button") que el usuario puede hacer clic para iniciar la búsqueda. Se establece el texto del botón y se le da un estilo personalizado utilizando los parámetros del constructor. Además, se especifica que cuando se hace clic en el botón, se debe llamar al método "buscar_recetas" de la clase actual (que se define en otra parte del código).

Finalmente, se empacan los elementos del marco de búsqueda utilizando el método "pack" de los widgets y se los coloca en el lado izquierdo del marco utilizando el parámetro "side" con el valor "left".



```
#receta del dia
self.button_recipe_of_the_day = tk.Button(self.root, text="Receta del día", command=self.sho
self.button_recipe_of_the_day.pack(side="top", padx=0)

self.label_titulo = tk.Label(self.root, text="¡Encuentra la receta perfecta para hoy!", bg="
self.label_titulo.pack(fill=tk.BOTH, expand=True)

self.frame_recetas = tk.Frame(self.root)
self.frame_recetas.pack()

self.label_recetas = tk.Label(self.frame_recetas, text="Recetas:")
self.label_recetas.pack(side="left")

self.listbox_recetas = tk.Listbox(self.frame_recetas, width=30, height=10, font=("Arial", 12
self.listbox_recetas.pack(side="left")
self.listbox_recetas.configure(bg="black", fg="white")

self.frame_botones = tk.Frame(self.root)
self.frame_botones.pack(side="left")
```

Este código crea una ventana de GUI con una serie de elementos, como un botón para mostrar una receta aleatoria del día, una etiqueta de título, una lista de recetas, y un marco para los botones.

El botón "Receta del día" llama a una función "show_random_recipe" cuando se hace clic en él.

La etiqueta "¡Encuentra la receta perfecta para hoy!" se utiliza como un título para la ventana.

El marco "frame_recetas" contiene una etiqueta "Recetas:" y una lista de recetas "listbox_recetas".

El marco "frame_botones" se utiliza para contener los botones que se agregan más adelante en el código.

```
self.button_agregar = tk.Button(self.frame_botones, text="Agregar", command=self.agregar_rec
                                 borderwidth=2, relief=tk.RIDGE, bg='#ED1525', fg='wh
                                 font=('Arial Rounded MT Bold',12), padx=8, pady=5, c
self.button_agregar.pack(side="left")

self.button_modificar = tk.Button(self.frame_botones, text="Modificar ", command=self.modifi
                                 font=('Arial Rounded MT Bold',12), padx=8, pady=5, c
self.button_modificar.pack(side="left")

self.button_eliminar = tk.Button(self.frame_botones, text="Eliminar ", command=self.eliminar
                                 font=('Arial Rounded MT Bold',12), padx=8, pady=5, c
self.button_eliminar.pack(side="left")

self.button_mostrar = tk.Button(self.root, text="Mostrar", command=self.mostrar_receta,borde
                                 font=('Arial Rounded MT Bold',12), padx=8, pady=5, c
self.button_mostrar.pack(side="right")
```

El primer botón se llama "Agregar" y se crea en la variable self.button_agregar. Este botón tiene un texto que dice "Agregar" y llama a la función self.agregar_receta cuando se hace clic en él. El botón tiene un borde con un ancho de 2 y un relieve de tipo tk.RIDGE, así como un fondo rojo oscuro (bg='#ED1525'), un texto blanco (fg='white') y un tipo de letra en negrita (font=('Arial Rounded MT Bold',12)). El botón también tiene un relleno horizontal de 8 y vertical de 5 (padx=8, pady=5) y un cursor que cambia a una mano (cursor='hand2'). Finalmente, se empaqueta en el frame self.frame_botones y se muestra a la izquierda (pack(side="left")).

El segundo botón se llama "Modificar" y se crea en la variable self.button_modificar. Este botón tiene un texto que dice "Modificar" y llama a la función self.modificar_receta cuando se hace clic en él. El botón tiene un borde con un ancho de 2 y un relieve de tipo tk.RIDGE, así como un fondo rojo oscuro (bg='#ED1525'), un texto blanco (fg='white') y un tipo de letra en negrita (font=('Arial Rounded MT Bold',12)). El botón también tiene un relleno horizontal de 8 y vertical de 5 (padx=8, pady=5) y un cursor que cambia a una mano (cursor='hand2'). Al igual que el primer botón, se empaqueta en el frame self.frame_botones y se muestra a la izquierda (pack(side="left")).

El tercer botón se llama "Eliminar" y se crea en la variable self.button_eliminar. Este botón tiene un texto que dice "Eliminar" y llama a la función self.eliminar_receta cuando se hace clic en él. El botón tiene un borde con un ancho de 2 y un relieve de tipo tk.RIDGE, así como un fondo rojo oscuro (bg='#ED1525'), un texto blanco (fg='white') y un tipo de letra en negrita (font=('Arial Rounded MT Bold',12)). El botón también tiene un relleno horizontal de 8 y vertical de 5 (padx=8, pady=5) y un cursor que cambia a una mano (cursor='hand2'). Al igual que los dos botones anteriores, se empaqueta en el frame self.frame_botones y se muestra a la izquierda (pack(side="left")).

El cuarto botón se llama "Mostrar" y se crea en la variable self.button_mostrar. Este botón tiene un texto que dice "Mostrar" y llama a la función self.mostrar_receta cuando se hace clic en él. El botón tiene un borde con un ancho de 2 y un relieve de tipo tk.RIDGE, así como un fondo rojo oscuro (bg='#ED1525'), un texto blanco (fg='white') y un tipo de letra en negrita (font=('Arial Rounded MT Bold',12)). El botón también tiene un relleno horizontal de 8 y vertical de 5 (`padx=8, pady

```
self.root.protocol("WM_DELETE_WINDOW", self.guardar_recetas_en_csv)
```

La línea en cuestión establece una función para el evento "WM_DELETE_WINDOW", que ocurre cuando el usuario cierra la ventana de la aplicación. En este caso, la función "guardar_recetas_en_csv" se llamará automáticamente antes de que la ventana se cierre, lo que le permite a la aplicación guardar los datos de recetas en un archivo CSV antes de que la ventana se cierre.

```
# Agregar receta predeterminada
    receta_predeterminada = Receta(
        nombre="Tortilla española",
        ingredientes=[
            Ingrediente(nombre="Ingrediente 1", unidad_medida="gramos", cantidad=100),
            Ingrediente(nombre="Ingrediente 2", unidad_medida="tazas", cantidad=1),
        ],
        preparacion="Instrucciones de la receta predeterminada.",
        tiempo_preparacion=30,
        tiempo_coccion=60,
        fecha_creacion='2022-03-08'
    )
```

Este código crea una instancia de la clase Receta y la asigna a la variable "receta_predeterminada". La instancia de la clase Receta tiene varios atributos, incluyendo el nombre de la receta, una lista de ingredientes, instrucciones de preparación, tiempos de preparación y cocción y fecha de creación. En este caso, la receta predeterminada es una "Tortilla española" con dos ingredientes, "Ingrediente 1" y "Ingrediente 2", y una preparación y tiempos de preparación y cocción predeterminados. La fecha de creación de la receta es el 8 de marzo de 2022.

```
self.recetas.append(receta_predeterminada2)
self.listbox_recetas.insert(tk.END, receta_predeterminada2.nombre)
```

Este código agrega una receta predeterminada a una lista de recetas y la inserta en un cuadro de lista en la interfaz gráfica de usuario. En particular, self.recetas es una lista de objetos de la clase Receta y receta_predeterminada2 es una instancia de la clase Receta que ha sido previamente definida con valores predeterminados para sus atributos, incluyendo su nombre y sus ingredientes. self.recetas.append(receta_predeterminada2) agrega esta receta a la lista self.recetas, mientras que self.listbox_recetas.insert(tk.END, receta_predeterminada2.nombre) inserta el nombre de la receta en un cuadro de lista de la interfaz gráfica de usuario.

```
self.root.title("RecetasYa")
```

Este código establece el título de la ventana principal de la aplicación en "RecetasYa". "self.root" hace referencia a la instancia de la clase "Tk" que representa la ventana principal de la aplicación. "title" es un método de la clase "Tk" que permite establecer el título de la ventana.

RecetasYa

¡no pidas delivery, cociná!

Por más información
comunicate con **Agustin Vallejo**
aguvallejo18@gmail.com