Universidad de San Carlos de Guatemala
Escuela de Ingeniería en Ciencias y Sistemas, Facultad de Ingeniería
Introducción a la programación y computación 2, 2do. Semestre 2023.

ENCRIPTACION DE MENSAJES MEDIANTE EL MANEJO DE DRONES

202201947 – Pablo Andres Rodriguez Lima

Resumen

El programa establecido es una versión de manejos de listas y una interfaz gráfica mediante la cual puede ingresar drones y cada uno tiene contenido y dependiendo de las instrucciones puede manejarlas y desvelar un mensaje.

El manejo de una interfaz grafica es una de las maneras mas cómodas de trabajar un proyecto mas como es solicitado para el manejo de software.

Hay que considerar que al manejar listas en un lenguaje de programación puede beneficiar ya que se pueden crear tablas tipo matrices anidando listas lo que puede beneficiar el manejo de datos.

Palabras clave

1. Clase
2. Objeto
3. Método
4. Nodo
5. Lista enlazada doble

Abstract

The established program is a version of list management and a graphical interface through which you can input drones, each of which has content, and depending on the instructions, can handle them and reveal a message.

Managing a graphical interface is one of the most convenient ways to work on a project, especially as it is requested for software management.

It is important to consider that managing lists in a programming language can be beneficial since you can create table-like matrices by nesting lists, which can aid in data management.

Keywords

1. Class
2. Object
3. Method
4. Node
5. Doubly Linked List

Introducción

Este proyecto tiene como objetivo fundamental la creación de una innovadora tecnología para el envío de mensajes encriptados, garantizando su seguridad frente a interceptaciones y desciframientos por parte de personal o instituciones no autorizadas.

El sistema diseñado consta de dos componentes clave: un componente emisor de mensajes y un componente receptor de mensajes expuestos en un archivo tipo XML.

Se presenta un listado de temas los cuales pueden ser de beneficio para entender el proyecto y un esquema del tipo diagrama de clases para entender el manejo del proyecto.

Desarrollo del tema

1. ¿Qué es Python?

Python es un lenguaje de programación ampliamente utilizado en las aplicaciones web, el desarrollo de software, la ciencia de datos y el machine learning (ML). Los desarrolladores utilizan Python porque es eficiente y fácil de aprender, además de que se puede ejecutar en muchas plataformas diferentes. El software Python se puede descargar gratis, se integra bien a todos los tipos de sistemas y aumenta la velocidad del desarrollo.

1.1) ¿Cómo se utiliza Python?

El lenguaje Python se aplica a varios casos de uso en el desarrollo de aplicaciones, incluidos los ejemplos siguientes:

Desarrollo web del lado del servidor

El desarrollo web del lado del servidor incluye las funciones complejas de backend que los sitios web llevan a cabo para mostrar información al usuario. Por ejemplo, los sitios web deben interactuar con las bases de datos, comunicarse con otros sitios web y proteger los datos cuando se los envía a través de la red.

Python es útil para escribir código del lado del servidor debido a que ofrece muchas bibliotecas que constan de código prescrito para crear funciones de backend complejas. Los desarrolladores también utilizan un amplio rango de marcos de Python que proporcionan todas las herramientas necesarias para crear aplicaciones web con mayor rapidez y facilidad. Por ejemplo, los desarrolladores pueden crear la aplicación web esqueleto en segundos porque no deben escribirla desde cero. Pueden probarla por medio de las herramientas de prueba del marco, sin depender de herramientas de prueba externas.

1.2) Automatización con scripts de Python

Un lenguaje de scripting es un lenguaje de programación que automatiza las tareas que suelen llevar a cabo las personas. Los programadores utilizan ampliamente los scripts de Python para automatizar muchas tareas diarias.

1.3) XML tree and elements

XML es un formato de datos inherentemente jerárquico, y la forma más natural de representarlo es con un árbol. ET tiene dos clases para este propósito: ElementTree representa todo el documento XML como un árbol, y Element representa un solo nodo en este árbol. Las interacciones con todo el documento (lectura y escritura desde/hacia archivos) generalmente se realizan en el nivel de ElementTree. Las interacciones con un solo elemento XML y sus subelementos se realizan en el nivel de Element.

Implementaciones para resolución del proyecto:

a) Clases con métodos y atributos.
b) Clases del tipo nodo.
c) Clases del tipo listas para almacenar datos.

a) Clases con métodos y atributos

Una clase define una plantilla o molde para crear objeto, los cuales son instancias de esa clase. Los objetos creados a partir de una clase tienen las mismas propiedades y comportamientos definidos por la clase, pero pueden tener valores diferentes para los atributos que se definen en la clase.

En Python, una clase se define mediante la palabra clave «class», seguida del nombre de la clase y dos puntos (:) y luego el cuerpo de la clase. El cuerpo de la clase contiene definiciones de métodos y atributos, que pueden ser públicos o privados según su acceso.

Un atributo es una variable que se define dentro de una clase, la cual almacena datos que pertenecen a un objeto de esa clase. Los atributos se utilizan para representar características o propiedades de un objeto, como su estado actual, su identificador, su tamaño, su color, etc.

Los atributos pueden ser de diferentes tipos de datos, como enteros, flotantes, cadenas, listas, diccionarios, entre otros. Además, los atributos pueden tener distintos niveles de visibilidad, que se especifican mediante los modificadores de acceso en la definición de la clase. Por defecto, los atributos son públicos en Python, lo que significa que puede accederse a ellos desde cualquier lugar del programa.

En la definición de una clase, los atributos se definen como variables que se inicializan en el método especial __init__. Por ejemplo, en la clase Persona que definimos anteriormente, los atributos «nombre" y «edad» se definen de la siguiente manera:

b) Clases del tipo nodo.

Que es un nodo:
El nodo es un elemento de la lista el cual contiene: el elemento que queremos guardar, una referencia hacia el nodo anterior y una referencia al nodo siguiente.

Nodo de una lista.
Comenzaremos con construir un nodo utilizando una clase class Nodo: el constructor toma de valores a self y a elemento el cual será el elemento que se encuentre dentro del nodo, además, podemos ver que se encuentran otros dos atributos los cuales nos van a indicar las conexiones entre los nodos. Inicializamos siguiente y anterior a None ya al crear un Nodo queremos que no apunten a ningún otro nodo. Y por último tenemos al atributo elemento el cual será el dato que vamos a guardar en el nodo.

Clase y constructor de un Nodo.
Constructor de la lista doblemente ligada.
A continuación, crearemos una clase para la lista doblemente ligada class doubleList: la cual contiene únicamente el nodo inicial o el root de la lista el cual le da un inicio a la lista. Inicializamos root hacia None ya que al crear una lista queremos que este vacía. Durante el artículo crearemos varios métodos para interactuar con la lista, asegúrate de agregar esos métodos a esta clase.

c) Clases del tipo listas para almacenar datos.

Clase y constructor de una lista doblemente ligada.
¿Como se ve una lista cuyo root apunta al nodo?

La lista doblemente ligada inicia siempre con un root el cual va a apuntar al primer nodo de la lista.

Root referenciando al primer nodo de una lista.

Funciones para insertar un elemento a la lista doblemente ligada.

Insertando un elemento a una lista doblemente ligada vacía.

Para esta función tomaremos como parámetro a self y dato. Dato será el valor que va a tomar el elemento del nodo que vamos a insertar. Nuestra primera función será la siguiente: la cual va a recibir como parámetro a self y a dato el cual será el elemento que queremos guardar en el nodo. Al insertar nuestro primer elemento a la lista primero queremos revisar que la lista este vacía, si está vacía vamos a crear el primer nodo y pasaremos como parámetro el dato de la clase. En caso contrario vamos a desplegar un mensaje al usuario de que la lista no está vacía.

2. ¿Qué es Graphviz?

Es un conjunto de herramientas open-source realizado inicialmente en los laboratorios de investigación de AT&T para el dibujo de gráficos especificados en lenguaje de scripts DOT. Provee librerías para ser usadas por otras aplicaciones. Graphviz es software libre licenciado bajo CPL (Common Public License).

2.1 Partes de una interfaz Gráfica.

Tkinter es una librería del lenguaje de programación Python y funciona para la creación y el desarrollo de aplicaciones de escritorio. Esta librería facilita el posicionamiento y desarrollo de una interfaz gráfica de escritorio con Python. Tkinter es el paquete estándar de Python para interactuar con Tkt.

Tkinter es el paquete más utilizado para crear interfaces gráficas en Python. Es una capa orientada a objetos basada en Tcl (sencillo y versátil lenguaje de programación open-source) y Tk (la herramienta GUI estándar para Tcl).

2.2 Widgets

A la hora de montar una vista con Tkinter, nos basaremos en widgets jerarquizados, que irán componiendo poco a poco nuestra interfaz. Algunos de los más comunes son:

Tk: es la raíz de la interfaz, donde vamos a colocar el resto de widgets.

Frame: marco que permite agrupar diferentes widgets.

Label: etiqueta estática que permite mostrar texto o imagen.

Entry: etiqueta que permite introducir texto corto (típico de formularios).

Text: campo que permite introducir texto largo (típico para añadir comentarios).

Button: ejecuta una función al ser pulsado.

Radiobutton: permite elegir una opción entre varias.

Checkbutton: permite elegir varias de las opciones propuestas.

Menu: clásico menú superior con opciones (Archivo, Editar…).

Dialogs: ventana emergente (o pop-up).

Cuando vayamos a inicializar el componente, debemos pasar por constructor el elemento que quede “por encima” en la jerarquía de la vista (si queremos colocar una label dentro de un frame, al construir la etiqueta le pasaremos el marco como argumento del constructor).

2.3 Configuración

Para configurar un widget, simplemente llamamos a .config() y pasamos los argumentos que queramos modificar. Algunas opciones son:

bg: modifica el color de fondo. Se puede indicar con el color en inglés (incluyendo modificadores, como "darkgreen") o su código RGB en hexadecimal ("#aaaaaa" para blanco). Ojo: en MacOS no se puede modificar el color de fondo de los botones; aunque indiquemos un nuevo color, se mostrará en blanco. Lo más parecido que podemos hacer es configurar el highlightbackground, que pintará el fondo alrededor del botón del color que indiquemos.

fg: cambia el color del texto.

cursor: modifica la forma del cursor. Algunos de los más utilizados son "gumby", "pencil", "watch" o "cross".

height: altura en líneas del componente.

width: anchura en caracteres del componente.

font: nos permite especificar, en una tupla con nombre de la fuente, tamaño y estilo, la fuente a utilizar en el texto del componente. Por ejemplo, Font("Times New Roman", 24, "bold underline").

bd: modificamos la anchura del borde del widget.

relief: cambiamos el estilo del borde del componente. Su valor puede ser "flat", "sunken", "raised", "groove", "solid" o "ridge".

state: permite deshabilitar el componente (state=DISABLED); por ejemplo, una Label en la que no se puede escribir o un Button que no se puede clickar.

padding: espacio en blanco alrededor del widget en cuestión.

command: de cara a que los botones hagan cosas, podemos indicar qué función ejecutar cuando se haga click en el mismo.

2.4 Gestión de la composición

Es muy importante que, cuando tengamos configurado el componente, utilicemos un gestor de geometría de componentes. Si no, el widget quedará creado, pero no se mostrará.

Conclusiones

El proyecto tiene como vitalidad el uso correcto del manejo de los drones de tal manera que cada uno pueda tener su propio contenido.

Cada dron puede tener sus instrucciones de subir y bajar de tal manera que pueda ser mas optimo los tiempos de rastreo de mensajes.

La interfaz grafica puede ayudar y que el uso se de una manera mas intuitiva ayudando que todo sea mas amigable con el usuario.

Referencias bibliográficas

Gomez, O. A. (s/f). Visualizando Grafos usando Graphviz. OSiUX. Recuperado el 4 de septiembre de 2023, de https://osiux.com/visualizando-grafos-graphviz.org

Londoño, P. (2023, abril 26). Qué son las clases en Python, para qué sirven y cómo funcionan. Hubspot.es. https://blog.hubspot.es/website/clases-python

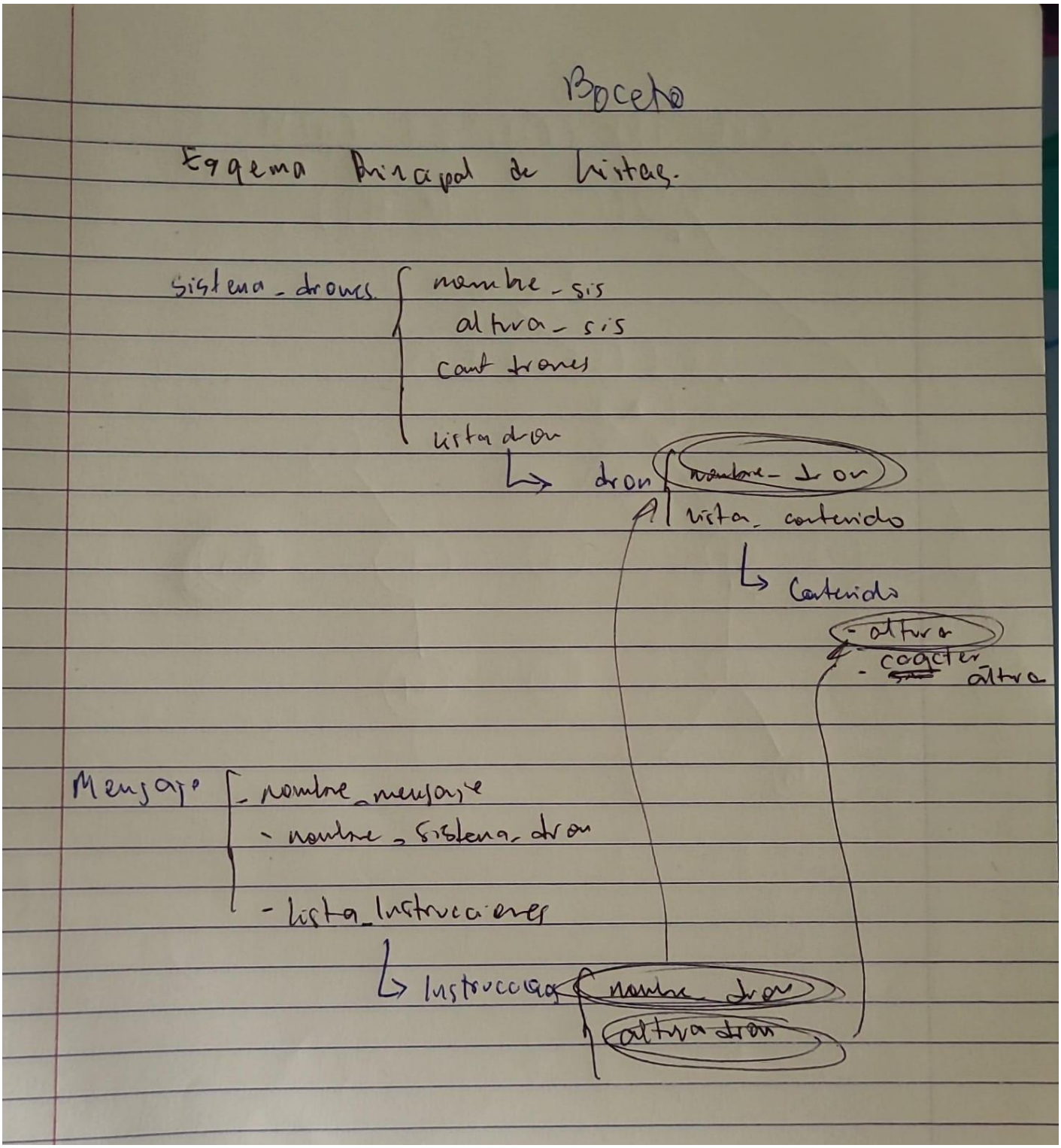
Pérez, J. R. C. (2020, noviembre 21). Implementando una lista doblemente ligada en Python. Medium.

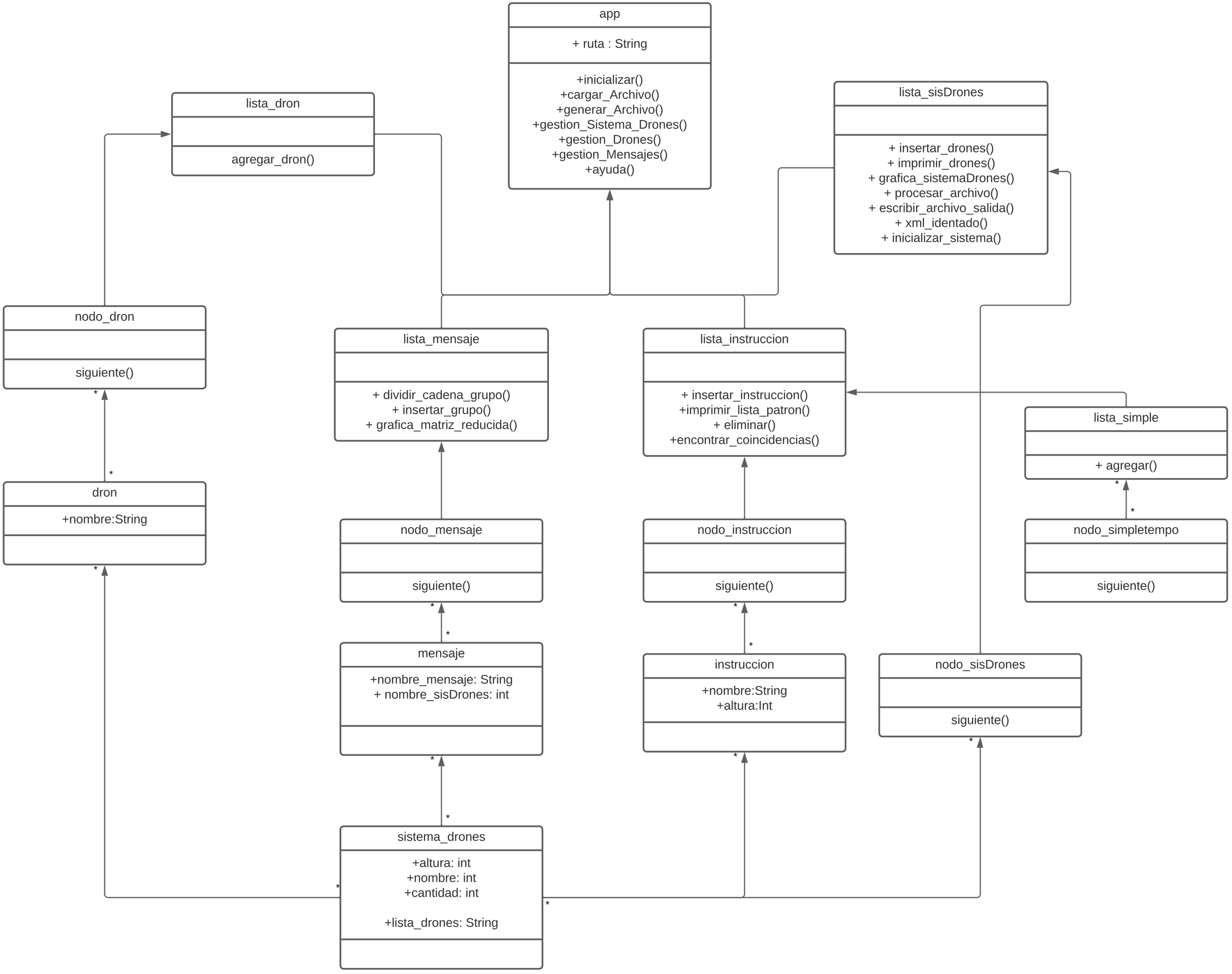
https://a01153884.medium.com/implementando-una-lista-doblemente-ligada-en-python-b8def2b8df73

xml.etree.ElementTree — The ElementTree XML API. (s/f). Python documentation. Recuperado el 4 de septiembre de 2023, de https://docs.python.org/3/library/xml.etree.elementtree.html

(S/f). Amazon.com. Recuperado el 4 de septiembre de 2023, de https://aws.amazon.com/es/what-is/python/

APENDICES


Boceto
Esquema Principal de Listas.
sistema_drones
nombre_sis
altura_sis
cant drones
lista dron
dron
nombre_dron
lista_contenido
Contenido
- altura
- caracter
altura
Mensaje
- nombre_mensaje
- nombre_sistema_dron
- lista_instrucciones
Instruccion
nombre_dron
altura dron


app
+ ruta : String
+inicializar()
+cargar_Archivo()
+generar_Archivo()
+gestion_Sistema_Drones()
+gestion_Drones()
+gestion_Mensajes()
+ayuda()
lista_dron
agregar_dron()
lista_sisDrones
+ insertar_drones()
+ imprimir_drones()
+ grafica_sistemaDrones()
+ procesar_archivo()
+ escribir_archivo_salida()
+ xml_identado()
+ inicializar_sistema()
nodo_dron
siguiente()
lista_mensaje
+ dividir_cadena_grupo()
+ insertar_grupo()
+ grafica_matriz_reducida()
lista_instruccion
+ insertar_instruccion()
+imprimir_lista_patron()
+ eliminar()
+encontrar_coincidencias()
lista_simple
+ agregar()
dron
+nombre:String
nodo_mensaje
siguiente()
nodo_instruccion
siguiente()
nodo_simpletempo
siguiente()
mensaje
+nombre_mensaje: String
+ nombre_sisDrones: int
instruccion
+nombre:String
+altura:Int
nodo_sisDrones
siguiente()
sistema_drones
+altura: int
+nombre: int
+cantidad: int
+lista_drones: String
*