documentación de la implementación

Presly Javier Romero Coll

Universidad Del Norte

Ingeniería De Sistemas

Tabla de contenido

[Introducción 2](#_Toc135425577)

[Descripción General 2](#_Toc135425578)

[Clases de la implementación 2](#_Toc135425579)

[Métodos y Funcionamiento 3](#_Toc135425580)

[Patrón de Diseño Implementado: Singleton 4](#_Toc135425581)

[Bibliografía 5](#_Toc135425582)

## Introducción

Este informe documenta la implementación de un sistema de gestión de clasificación de residuos en Python. El sistema permite organizar y controlar las rutas de recolección, así como clasificar y registrar los residuos en un centro especializado. Se detallarán las clases, métodos y su funcionamiento en el código, destacando la implementación del patrón Singleton para garantizar la existencia de un único centro de clasificación. Además, se presentarán los resultados del plan de pruebas realizado para verificar la funcionalidad del sistema. Este informe proporciona una visión completa y detallada de la implementación, sirviendo como guía para comprender su estructura y uso.

## Descripción General

La aplicación de clasificación de residuos es un sistema que permite gestionar el proceso de recolección y clasificación de diferentes tipos de residuos. Está desarrollada en Python utilizando el paradigma de Programación Orientada a Objetos (POO), lo que proporciona una estructura modular y reutilizable para el código.

El sistema consta de las siguientes clases principales:

## Clases de la implementación

1. **Equipo**: Representa un equipo de trabajo encargado de la recolección de residuos. Almacena información sobre el conductor y los asistentes del equipo.

2. **Camión**: Representa un camión de recolección de residuos. Contiene atributos como la placa del camión, el equipo asignado, la ruta y el día de la recolección.

3. **Punto**: Representa un punto geográfico en el sistema. Almacena las coordenadas de longitud y latitud.

4. **PuntoRecoleccion**: Representa un punto de recolección de residuos. Contiene una lista de puntos geográficos y la clasificación de residuos asociada a ese punto.

5. **Ruta**: Representa una ruta de recolección de residuos. Almacena información sobre el punto de origen, los puntos de recolección, el punto de clasificación, las horas de inicio y fin, y el día de la ruta. También registra la cantidad de clasificación de residuos por turno.

6. **Clasificación**: Representa la clasificación de residuos en diferentes categorías, como vidrio, papel, plástico, metal y orgánico. Inicialmente, todas las categorías están configuradas en cero.

7. **LugarClasificacion**: Implementa el patrón de diseño Singleton para asegurar que solo haya un centro de clasificación de residuos. Almacena la clasificación total de residuos recibidos.

## Métodos y Funcionamiento

**Clase Equipo**

* `\_\_init\_\_(self, conductor\_nombre, conductor\_documento, asistente1\_nombre, asistente1\_documento, asistente2\_nombre, asistente2\_documento)`: Constructor de la clase Equipo que inicializa los atributos del equipo con los datos proporcionados.

**Clase Camión**

* `\_\_init\_\_(self, placa, equipo, ruta, dia)`: Constructor de la clase Camion que inicializa los atributos del camión con los datos proporcionados.

**Clase Punto**

* `\_\_init\_\_(self, longitud, latitud)`: Constructor de la clase Punto que inicializa los atributos de longitud y latitud del punto.

**Clase PuntoRecoleccion**

* `\_\_init\_\_(self, puntos, clasificacion)`: Constructor de la clase PuntoRecoleccion que inicializa la lista de puntos y la clasificación asociada al punto.

**Clase Ruta**

* `\_\_init\_\_(self, punto\_origen, puntos\_recoleccion, punto\_clasificacion, hora\_inicio, hora\_fin, dia)`: Constructor de la clase Ruta que inicializa los atributos de la ruta con los datos proporcionados.
* `actualizar\_cantidad\_clasificacion(self, turno, clasificacion)`: Método que actualiza la cantidad de clasificación de residuos por turno en la ruta.

**Clase Clasificación**

* `\_\_init\_\_(self, vidrio=0, papel=0, plastico=0, metal=0, organico=0)`: Constructor de la clase Clasificación que inicializa las categorías de clasificación de residuos con los valores proporcionados o cero por defecto.

**Clase LugarClasificacion**

* `\_\_new\_\_(cls, \*args, \*\*kwargs)`: Método que implementa el patrón de diseño Singleton, asegurando que solo exista una instancia de la clase LugarClasificacion.
* `\_\_init\_\_(self)`: Constructor de la clase LugarClasificacion que inicializa la clasificación total de residuos recibidos.
* `recibir\_residuos(self, clasificacion)`: Método que recibe la clasificación de residuos y actualiza la clasificación total almacenada.

## Patrón de Diseño Implementado: Singleton

En la implementación del código, se ha utilizado el patrón de diseño Singleton en la clase LugarClasificacion. Este patrón garantiza que solo exista una instancia de la clase en toda la aplicación. Al utilizar el método `\_\_new\_\_`, se controla la creación de instancias y se asegura que siempre se retorne la misma instancia de la clase LugarClasificacion.

Este patrón es especialmente útil en este contexto, ya que solo se requiere un centro de clasificación de residuos en el sistema y se desea evitar la creación de múltiples instancias de la clase.

# Bibliografía

Ionos. (s.f.). *Singleton: te explicamos cómo funciona el patrón singleton.* Obtenido de https://www.ionos.mx/digitalguide/paginas-web/desarrollo-web/patron-singleton/

El presente informe se benefició de las valiosas ayudas proporcionadas por el modelo de lenguaje de inteligencia artificial, desarrollado por OpenAI. Específicamente, se destaca la asistencia brindada en la implementación del código en Python y la explicación detallada de los patrones de diseño, como el patrón Singleton. La orientación precisa y clara del modelo permitió identificar oportunidades de mejora para el código y desarrollar un plan de pruebas efectivo. Estas contribuciones fueron fundamentales para lograr una implementación robusta y comprensiva del sistema de gestión de residuos.