

## **Proyecto de reciclaje**

Kevin A Herrera P.

Dilan F Hernández G.

Ingeniería de sistemas, Corporación Universitaria Minuto de Dios

74169: Programación Orientada a Objetos

Jasson Diaz Zamudio

2025

## **Justificación del Proyecto**

Este proyecto responde a la urgente necesidad de promover y optimizar la gestión del reciclaje, tanto a nivel familiar como comunitario. Su principal objetivo es fomentar una mayor conciencia ambiental, subrayando el papel crucial del reciclaje en la mitigación de la contaminación, la conservación de los recursos naturales y la reducción del volumen de desechos destinados a los vertederos. Además, la aplicación proporciona una herramienta efectiva para incentivar y supervisar la participación activa de las familias en prácticas de reciclaje. Esto se logra mediante un sistema de puntos que cuantifica el esfuerzo de reciclaje, permitiendo a los usuarios registrar y visualizar su contribución de manera tangible. El sistema también facilita la organización y el seguimiento detallado de los materiales reciclados por cada familia, lo que puede ser de gran utilidad para programas de reciclaje a escala municipal o comunitaria, permitiendo un análisis más preciso y la implementación de estrategias más efectivas. La simplicidad y accesibilidad son características fundamentales de la aplicación, que ofrece una interfaz intuitiva que simplifica el registro de materiales reciclables para los usuarios. Finalmente, el proyecto posee un notable potencial de escalabilidad, abriendo la puerta a futuras expansiones que podrían incluir la incorporación de una gama más amplia de materiales, la implementación de sistemas de recompensas para incentivar aún más la participación, o la integración con servicios de recolección de reciclaje existentes.

## Descripción de Clases

El proyecto se fundamenta en una arquitectura orientada a objetos, donde las clases se diseñan para estructurar y gestionar eficientemente los datos y la lógica de la aplicación:

### **MaterialReciclado (Reciclaje\_model.py):**

**Propósito:** Define el esquema base para representar cualquier material susceptible de ser reciclado.

#### **Atributos:**

tipo (str): Especifica el tipo de material (por ejemplo, "plástico", "vidrio").

peso (float): Almacena el peso del material en kilogramos.

fecha (datetime): Registra la fecha en que se realizó el registro del material.

#### **Métodos:**

`__init__(self, tipo, peso, fecha)`: Constructor que inicializa los atributos del objeto.

`calcular_puntos(self)`: Método que calcula los puntos otorgados por el material, basado en su peso (1 punto por kilogramo en la clase base).

### **Plastico, Vidrio, Papel (Reciclaje\_model.py):**

**Propósito:** Representan los tipos específicos de materiales reciclables, heredando las propiedades y comportamientos de la clase MaterialReciclado.

**Atributos:** Heredan los atributos tipo, peso y fecha de la clase MaterialReciclado.

**Métodos:**

calcular\_puntos(self): Implementan una versión especializada del cálculo de puntos, asignando valores diferenciados según el tipo de material:

Plastico: peso \* 2

Vidrio: peso \* 1.5

Papel: peso \* 1.2

**Familia (Reciclaje\_model.py):**

**Propósito:** Modela a una familia que participa en el programa de reciclaje, registrando sus actividades.

**Atributos:**

nombre (str): Identifica el nombre de la familia.

integrantes (int): Indica el número de miembros en la familia.

direccion (str): Especifica la dirección de residencia de la familia.

materiales (list): Contiene una lista de objetos MaterialReciclado que la familia ha registrado.

**Métodos:**

\_\_init\_\_(self, nombre, integrantes, direccion): Constructor que inicializa los atributos de la familia.

agregar\_material(self, material): Permite añadir un nuevo material reciclado a la lista de la familia.

obtener\_resumen(self): Genera un resumen de la actividad de reciclaje de la familia, incluyendo un desglose de los materiales y el total de puntos acumulados.

### **Controlador (Reciclaje\_controller.py):**

**Propósito:** Actúa como el componente central que coordina la lógica de la aplicación, gestionando la interacción entre los datos (modelo) y la interfaz de usuario (vista).

### **Atributos:**

familias (list): Mantiene una lista de objetos Familia que están registrados en el sistema.

Métodos:

\_\_init\_\_(self): Constructor que inicializa la lista de familias.

ejecutar(self): Implementa el bucle principal de la aplicación, presentando el menú de opciones al usuario y procesando sus selecciones.

registrar\_familia(self): Facilita el registro de una nueva familia en el sistema.

registrar\_material(self): Permite registrar un material reciclado para una familia existente.

mostrar\_resumenes(self): Presenta los resúmenes de reciclaje de todas las familias registradas.

### **Funciones en Reciclaje\_view.py:**

## Funciones:

`pedir_familia()`: Solicita al usuario los datos necesarios para registrar una familia.

pedir\_material(): Solicita al usuario los datos de un material reciclado.

mostrar\_resumen(familia): Presenta el resumen de reciclaje de una familia específica.

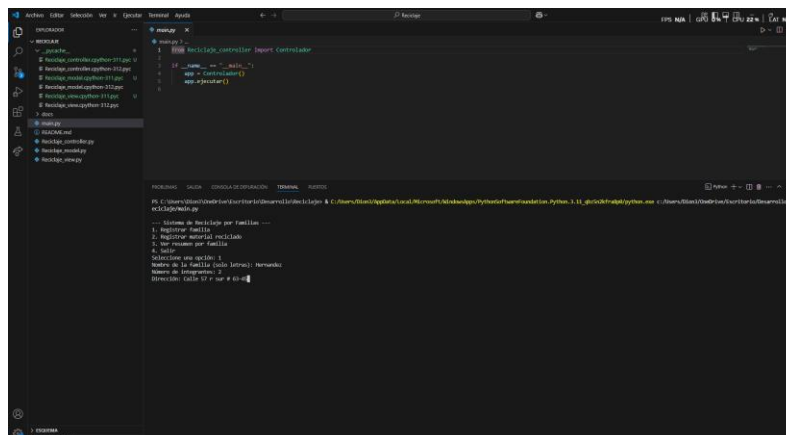
## Evidencias de ejecución y mapa de navegación del sistema

- 
- The screenshot displays a Windows 10 desktop environment. The primary focus is the Visual Studio Code (VS Code) editor, which is open to a file named `main.py` located within a project directory `reciclaje`. The Python code in `main.py` defines a `Controlador` class with three methods: `registrar`, `eliminar`, and `ver_resumen`. The `registrar` method takes a family name and a recycling material as input and adds them to a list. The `eliminar` method removes a family name from the list. The `ver_resumen` method prints the current state of the list. The terminal window at the bottom shows the command `python main.py` being executed, and the output prompt `Seleccione una opción:`  is visible.
- ```

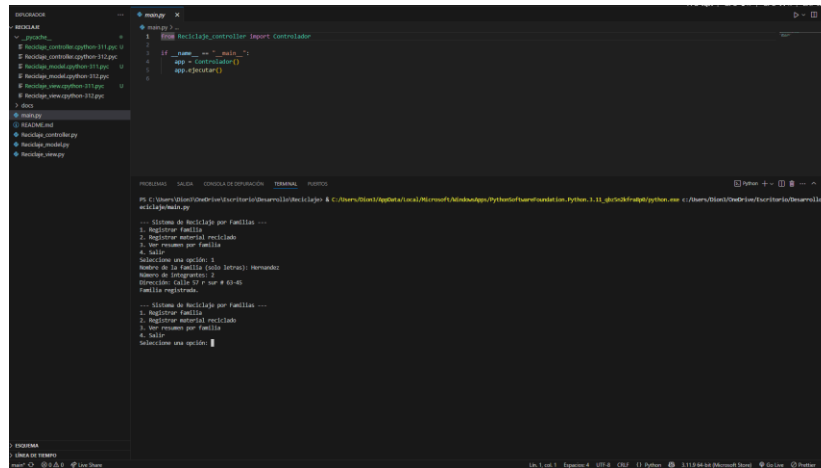
# main.py
from Reciclaje_controller import Controlador

1
2
3 if __name__ == "__main__":
4     app = Controlador()
5     app.registrar()
6

```
- PROBLEMAS CALDA CONTROL DE VERSIONES TERMINAL PUERTO
- P5 C:\Users\idreiv\OneDrive\Escritorio\Desarrollo\Reciclaje & C:\Users\idreiv\OneDrive\Local Microsoft\Microsoft\Python\Python\Python3.11.0\Scripts\python.exe C:\Users\idreiv\OneDrive\Escritorio\Desarrollo\Reciclaje\main.py
- Seleccione de Reciclaje por familia ---
1. Registrar familia
  2. Eliminar material reciclado
  3. Ver resumen por familia
  4. Salir
- Seleccione una opción:



## 5. Se finalizo el proceso



```
from __future__ import annotations
from typing import List, Optional, Union

from pydantic import BaseModel, Field, validator

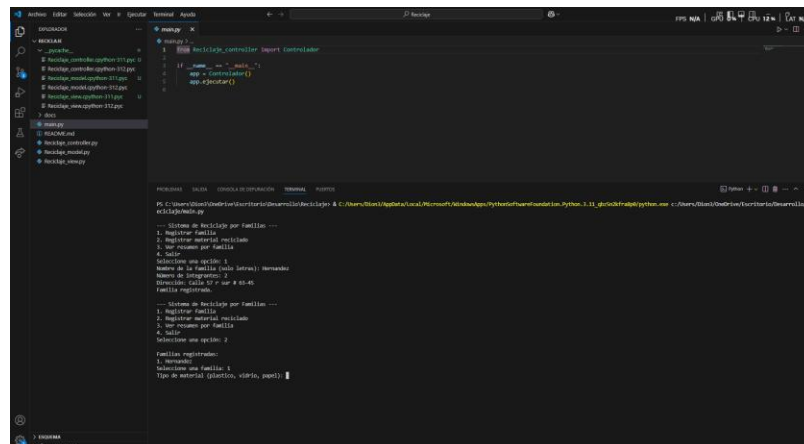
from reciclaje.controller import Controller

if __name__ == "__main__":
    app = Controller()
    app.execute()
```

```
--- Sistema de Reciclaje por Familias ---
1. Registrar familia
2. Registrar material reciclado
3. Ver resumen por familia
4. Salir
Seleccione una opción: 1
Nombre de la familia (solo letras): Hernandez
Número de integrantes: 2
Dirección (calle y n° por # 60-65):
Familia registrada.

--- Sistema de Reciclaje por Familias ---
1. Registrar familia
2. Registrar material reciclado
3. Ver resumen por familia
4. Salir
Seleccione una opción: 2
```

## 6. Seleccionamos la opción 2 para ingresar un material

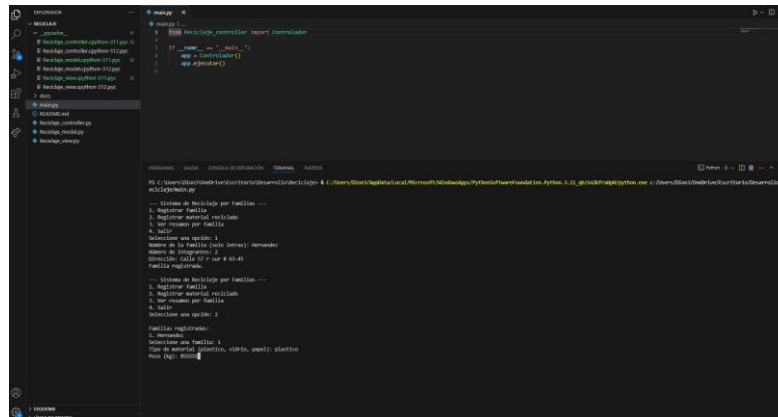


```
--- Sistema de Reciclaje por Familias ---
1. Registrar familia
2. Registrar material reciclado
3. Ver resumen por familia
4. Salir
Seleccione una opción: 1
Nombre de la familia (solo letras): Hernandez
Número de integrantes: 2
Dirección (calle y n° por # 60-65):
Familia registrada.

--- Sistema de Reciclaje por Familias ---
1. Registrar familia
2. Registrar material reciclado
3. Ver resumen por familia
4. Salir
Seleccione una opción: 2
Familia registrada:
1. Hernandez
Seleccione una familia: 1
Tipo de material (plástico, vidrio, papel):
```



## 7. Seleccionamos el tipo de material



```
from typing import List, Dict, Optional
from abc import ABC, abstractmethod
from dataclasses import dataclass
from datetime import datetime
from decimal import Decimal

class Reciclaje_Controller:
    """Controlador para el sistema de reciclaje"""

    def __init__(self):
        self._familias: List[Familia] = []
        self._materiales: List[Material] = []
        self._reciclados: List[Reciclado] = []

    def _mostrar_menu(self):
        """Muestra el menú de opciones"""
        print("\n--- Sistema de Reciclaje por Familias ---")
        print("1. Registrar familia")
        print("2. Registrar material reciclado")
        print("3. Ver resumen por familia")
        print("4. Salir")
        print("Seleccione una opción: ")

    def _seleccionar_opcion(self):
        """Selecciona una opción del menú"""
        while True:
            opcion = input()
            if opcion in ['1', '2', '3', '4']:
                return opcion
            else:
                print("Opción no válida. Intente de nuevo.")

    def _registrar_familia(self):
        """Registra una nueva familia"""
        print("Nombre de la familia (solo letras): ")
        nombre = input()
        print("Número de integrantes: ")
        integrantes = int(input())
        familia = Familia(nombre, integrantes)
        self._familias.append(familia)
        print(f"Familia registrada.")

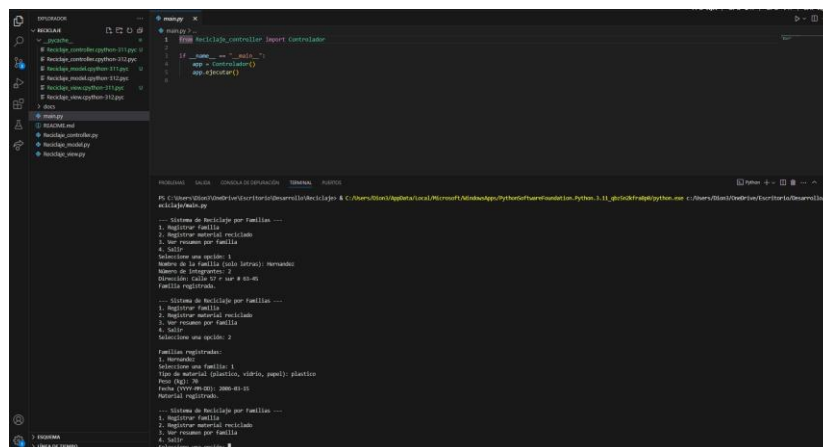
    def _registrar_material(self):
        """Registra un material reciclado"""
        print("Seleccione una familia: ")
        for i, familia in enumerate(self._familias):
            print(f"{i+1}. {familia.nombre} ({familia.integrantes} integrantes)")
        familia_id = int(input())
        if familia_id < 1 or familia_id > len(self._familias):
            print("Familia no encontrada.")
            return
        familia = self._familias[familia_id - 1]
        print("Tipo de material (plástico, vidrio, papel): ")
        tipo = input()
        print("Peso (kg): ")
        peso = float(input())
        material = Material(familia, tipo, peso)
        self._materiales.append(material)
        print("Material registrado.")

    def _ver_resumen(self):
        """Muestra el resumen de una familia"""
        print("Seleccione una familia: ")
        for i, familia in enumerate(self._familias):
            print(f"{i+1}. {familia.nombre} ({familia.integrantes} integrantes)")
        familia_id = int(input())
        if familia_id < 1 or familia_id > len(self._familias):
            print("Familia no encontrada.")
            return
        familia = self._familias[familia_id - 1]
        print(f"Resumen de materiales para familia: {familia.nombre}")
        print(f"Material: {familia.tipo}, Cantidad: {familia.peso} kg, Valor: {familia.valor}")
        print(f"Total de puntos: {familia.puntos}")

    def _ver_resumen_por_familia(self):
        """Muestra el resumen de todos los materiales"""
        print("Seleccione una opción: ")
        print("1. Registrar familia")
        print("2. Registrar material reciclado")
        print("3. Ver resumen por familia")
        print("4. Salir")
        opcion = self._seleccionar_opcion()
        if opcion == '1':
            self._registrar_familia()
        elif opcion == '2':
            self._registrar_material()
        elif opcion == '3':
            self._ver_resumen()
        elif opcion == '4':
            print("Programa finalizado.")
            exit()

    def _ver_resumen_por_familia(self):
        """Muestra el resumen de todos los materiales"""
        print("Seleccione una familia: ")
        for i, familia in enumerate(self._familias):
            print(f"{i+1}. {familia.nombre} ({familia.integrantes} integrantes)")
        familia_id = int(input())
        if familia_id < 1 or familia_id > len(self._familias):
            print("Familia no encontrada.")
            return
        familia = self._familias[familia_id - 1]
        print(f"Resumen de materiales para familia: {familia.nombre}")
        print(f"Material: {familia.tipo}, Cantidad: {familia.peso} kg, Valor: {familia.valor}")
        print(f"Total de puntos: {familia.puntos}")
```

## 8. Registramos el peso y la fecha



```
from typing import List, Dict, Optional
from abc import ABC, abstractmethod
from dataclasses import dataclass
from datetime import datetime
from decimal import Decimal

class Reciclaje_Controller:
    """Controlador para el sistema de reciclaje"""

    def __init__(self):
        self._familias: List[Familia] = []
        self._materiales: List[Material] = []
        self._reciclados: List[Reciclado] = []

    def _mostrar_menu(self):
        """Muestra el menú de opciones"""
        print("\n--- Sistema de Reciclaje por Familias ---")
        print("1. Registrar familia")
        print("2. Registrar material reciclado")
        print("3. Ver resumen por familia")
        print("4. Salir")
        print("Seleccione una opción: ")

    def _seleccionar_opcion(self):
        """Selecciona una opción del menú"""
        while True:
            opcion = input()
            if opcion in ['1', '2', '3', '4']:
                return opcion
            else:
                print("Opción no válida. Intente de nuevo.")

    def _registrar_familia(self):
        """Registra una nueva familia"""
        print("Nombre de la familia (solo letras): ")
        nombre = input()
        print("Número de integrantes: ")
        integrantes = int(input())
        familia = Familia(nombre, integrantes)
        self._familias.append(familia)
        print(f"Familia registrada.")

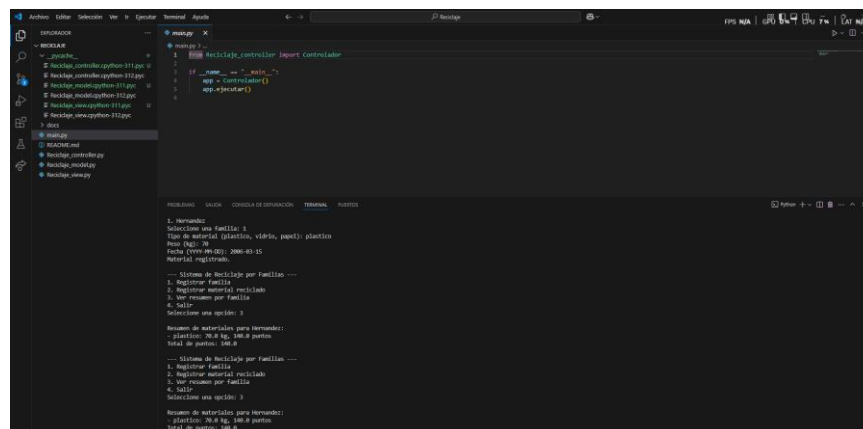
    def _registrar_material(self):
        """Registra un material reciclado"""
        print("Seleccione una familia: ")
        for i, familia in enumerate(self._familias):
            print(f"{i+1}. {familia.nombre} ({familia.integrantes} integrantes)")
        familia_id = int(input())
        if familia_id < 1 or familia_id > len(self._familias):
            print("Familia no encontrada.")
            return
        familia = self._familias[familia_id - 1]
        print("Tipo de material (plástico, vidrio, papel): ")
        tipo = input()
        print("Peso (kg): ")
        peso = float(input())
        material = Material(familia, tipo, peso)
        self._materiales.append(material)
        print("Material registrado.")

    def _ver_resumen(self):
        """Muestra el resumen de una familia"""
        print("Seleccione una familia: ")
        for i, familia in enumerate(self._familias):
            print(f"{i+1}. {familia.nombre} ({familia.integrantes} integrantes)")
        familia_id = int(input())
        if familia_id < 1 or familia_id > len(self._familias):
            print("Familia no encontrada.")
            return
        familia = self._familias[familia_id - 1]
        print(f"Resumen de materiales para familia: {familia.nombre}")
        print(f"Material: {familia.tipo}, Cantidad: {familia.peso} kg, Valor: {familia.valor}")
        print(f"Total de puntos: {familia.puntos}")

    def _ver_resumen_por_familia(self):
        """Muestra el resumen de todos los materiales"""
        print("Seleccione una opción: ")
        print("1. Registrar familia")
        print("2. Registrar material reciclado")
        print("3. Ver resumen por familia")
        print("4. Salir")
        opcion = self._seleccionar_opcion()
        if opcion == '1':
            self._registrar_familia()
        elif opcion == '2':
            self._registrar_material()
        elif opcion == '3':
            self._ver_resumen()
        elif opcion == '4':
            print("Programa finalizado.")
            exit()

    def _ver_resumen_por_familia(self):
        """Muestra el resumen de todos los materiales"""
        print("Seleccione una familia: ")
        for i, familia in enumerate(self._familias):
            print(f"{i+1}. {familia.nombre} ({familia.integrantes} integrantes)")
        familia_id = int(input())
        if familia_id < 1 or familia_id > len(self._familias):
            print("Familia no encontrada.")
            return
        familia = self._familias[familia_id - 1]
        print(f"Resumen de materiales para familia: {familia.nombre}")
        print(f"Material: {familia.tipo}, Cantidad: {familia.peso} kg, Valor: {familia.valor}")
        print(f"Total de puntos: {familia.puntos}")
```

## 9. Seleccionamos el 3 para ver el resumen de la familia



```
from typing import List, Dict, Optional
from abc import ABC, abstractmethod
from dataclasses import dataclass
from datetime import datetime
from decimal import Decimal

class Reciclaje_Controller:
    """Controlador para el sistema de reciclaje"""

    def __init__(self):
        self._familias: List[Familia] = []
        self._materiales: List[Material] = []
        self._reciclados: List[Reciclado] = []

    def _mostrar_menu(self):
        """Muestra el menú de opciones"""
        print("\n--- Sistema de Reciclaje por Familias ---")
        print("1. Registrar familia")
        print("2. Registrar material reciclado")
        print("3. Ver resumen por familia")
        print("4. Salir")
        print("Seleccione una opción: ")

    def _seleccionar_opcion(self):
        """Selecciona una opción del menú"""
        while True:
            opcion = input()
            if opcion in ['1', '2', '3', '4']:
                return opcion
            else:
                print("Opción no válida. Intente de nuevo.")

    def _registrar_familia(self):
        """Registra una nueva familia"""
        print("Nombre de la familia (solo letras): ")
        nombre = input()
        print("Número de integrantes: ")
        integrantes = int(input())
        familia = Familia(nombre, integrantes)
        self._familias.append(familia)
        print(f"Familia registrada.")

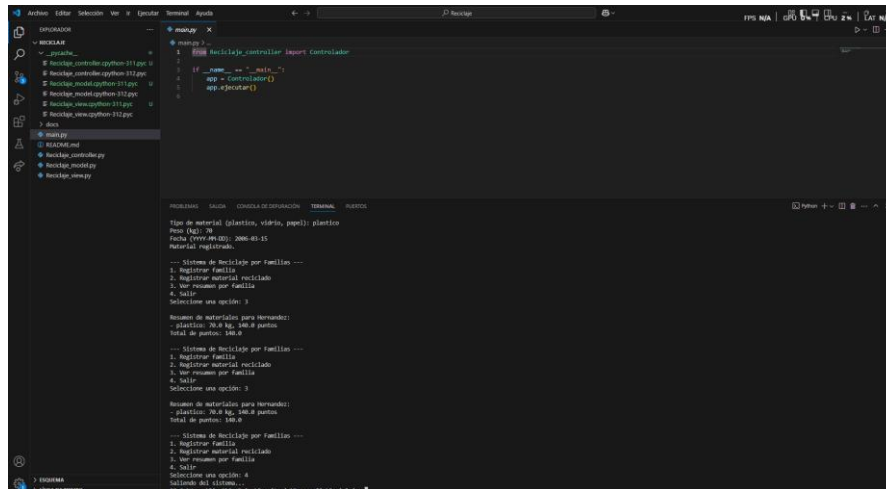
    def _registrar_material(self):
        """Registra un material reciclado"""
        print("Seleccione una familia: ")
        for i, familia in enumerate(self._familias):
            print(f"{i+1}. {familia.nombre} ({familia.integrantes} integrantes)")
        familia_id = int(input())
        if familia_id < 1 or familia_id > len(self._familias):
            print("Familia no encontrada.")
            return
        familia = self._familias[familia_id - 1]
        print("Tipo de material (plástico, vidrio, papel): ")
        tipo = input()
        print("Peso (kg): ")
        peso = float(input())
        material = Material(familia, tipo, peso)
        self._materiales.append(material)
        print("Material registrado.")

    def _ver_resumen(self):
        """Muestra el resumen de una familia"""
        print("Seleccione una familia: ")
        for i, familia in enumerate(self._familias):
            print(f"{i+1}. {familia.nombre} ({familia.integrantes} integrantes)")
        familia_id = int(input())
        if familia_id < 1 or familia_id > len(self._familias):
            print("Familia no encontrada.")
            return
        familia = self._familias[familia_id - 1]
        print(f"Resumen de materiales para familia: {familia.nombre}")
        print(f"Material: {familia.tipo}, Cantidad: {familia.peso} kg, Valor: {familia.valor}")
        print(f"Total de puntos: {familia.puntos}")

    def _ver_resumen_por_familia(self):
        """Muestra el resumen de todos los materiales"""
        print("Seleccione una opción: ")
        print("1. Registrar familia")
        print("2. Registrar material reciclado")
        print("3. Ver resumen por familia")
        print("4. Salir")
        opcion = self._seleccionar_opcion()
        if opcion == '1':
            self._registrar_familia()
        elif opcion == '2':
            self._registrar_material()
        elif opcion == '3':
            self._ver_resumen()
        elif opcion == '4':
            print("Programa finalizado.")
            exit()

    def _ver_resumen_por_familia(self):
        """Muestra el resumen de todos los materiales"""
        print("Seleccione una familia: ")
        for i, familia in enumerate(self._familias):
            print(f"{i+1}. {familia.nombre} ({familia.integrantes} integrantes)")
        familia_id = int(input())
        if familia_id < 1 or familia_id > len(self._familias):
            print("Familia no encontrada.")
            return
        familia = self._familias[familia_id - 1]
        print(f"Resumen de materiales para familia: {familia.nombre}")
        print(f"Material: {familia.tipo}, Cantidad: {familia.peso} kg, Valor: {familia.valor}")
        print(f"Total de puntos: {familia.puntos}")
```

10. Seleccionamos el 4 para salir



```
1 from abc import ABC, abstractmethod
2 from datetime import datetime
3
4 class Material:
5     def __init__(self, tipo, peso, fecha_registro):
6         self.tipo = tipo
7         self.peso = peso
8         self.fecha_registro = fecha_registro
9
10    def __str__(self):
11        return f"Material: {self.tipo}, {self.peso} kg, {self.fecha_registro}"
12
13    @abstractmethod
14    def registrar(self):
15        pass
16
17    @abstractmethod
18    def listar(self):
19        pass
20
21    @abstractmethod
22    def eliminar(self):
23        pass
24
25    @abstractmethod
26    def actualizar(self):
27        pass
28
29 class MaterialPlastico(Material):
30     def registrar(self):
31         print("Registrar Material Plastico")
32         peso = input("Peso (kg): ")
33         fecha_registro = input("Fecha (YYYY-MM-DD): ")
34         self.peso = float(peso)
35         self.fecha_registro = datetime.strptime(fecha_registro, "%Y-%m-%d")
36         print("Material registrado.")
37
38     def listar(self):
39         print("Listar Material Plastico")
40         print("----- Sistema de Reciclaje por Familias -----")
41         print("1. Registrar familia")
42         print("2. Registrar material reciclado")
43         print("3. Ver resumen por familia")
44         print("4. Salir")
45         print("Seleccione una opción: ")
46
47     def eliminar(self):
48         print("Eliminar Material Plastico")
49         print("Resumen de materiales para eliminar:")
50         print("plastico: 70.0 kg, 100.0 puntos")
51         print("Total de puntos: 100.0")
52
53     def actualizar(self):
54         print("Actualizar Material Plastico")
55         print("----- Sistema de Reciclaje por Familias -----")
56         print("1. Registrar familia")
57         print("2. Registrar material reciclado")
58         print("3. Ver resumen por familia")
59         print("4. Salir")
60         print("Seleccione una opción: ")
61
62 if __name__ == "__main__":
63     app = MaterialPlastico()
64     app.registrar()
65     app.listar()
66     app.eliminar()
67     app.actualizar()
```

## Conclusiones

En conclusión, este proyecto establece una base sólida y funcional para el registro y gestión de información relacionada con el reciclaje a nivel familiar. La implementación de la programación orientada a objetos proporciona una estructura de código bien organizada y facilita la futura expansión del sistema con nuevas funcionalidades o adaptaciones. La adopción del patrón Modelo-Vista-Controlador (MVC) promueve una clara separación de responsabilidades, lo que resulta en un código más limpio, modular y fácil de mantener. Si bien el sistema de puntos actual ofrece una forma sencilla de cuantificar el esfuerzo de reciclaje, su eficacia podría incrementarse mediante la introducción de esquemas de puntuación más sofisticados o la implementación de un sistema de recompensas que incentive aún más la participación. La interfaz de usuario, basada en texto, cumple con su función, pero podría mejorarse significativamente mediante el desarrollo de una interfaz gráfica que aumente su atractivo visual y facilidad de uso. Para mejorar la robustez y la integridad de los datos, se recomienda la

implementación de mecanismos de persistencia de datos que permitan guardar la información de forma permanente, así como la incorporación de validaciones y manejo de errores más exhaustivos. En resumen, este proyecto constituye un excelente punto de partida para el desarrollo de un sistema de gestión de reciclaje completo y adaptable, con un diseño que permite su evolución y adaptación a las cambiantes necesidades de los usuarios y las comunidades.