Sistema de Registro de Ingresos y Egresos

Este documento describe, de forma accesible, la aplicación escrita en Python que consta de tres archivos principales —model.py, storage.py y cli.py— explicando qué hace cada uno, cómo se relacionan y qué se necesita para instalarlos y utilizarlos.

1. Cómo instalar, editar y ejecutar la aplicación

Instalar Python 3.8 o superior desde https://python.org o el gestor de paquetes de tu sistema.

Descargar/copiar los tres archivos (model.py, storage.py, cli.py) en una carpeta, por ejemplo proyecto_cobros/.

La carpeta data/ se creará automáticamente la primera vez que ejecutes la app.

Ejecutar la aplicación de escritorio:

- python cli.py

2. model.py – Plantillas de datos

model.py define tres clases simples (sin lógica interna compleja) que funcionan como formularios de datos. Cada objeto se transforma más tarde en una tupla de texto para guardarse en disco.

Clase	Para qué sirve	Principales atributos
cobro	Representa un ingreso (cobro) registrado al condominio.	id, fecha, nombreCompleto, numParcela, imputacion1-3 (códigos), concepto1-3 (descripciones), importeBruto1-3, numCuentaA/B, montoA/B, impuestoDBCRb, anticipoIIBB, iva, observaciones (21 campos en total).
pago	Representa un egreso (pago) efectuado.	id, fecha, razonSocial, concepto, tipoComprobante, numCuenta, montoNeto, iva, cuentaAcreditar, impuestoDBCRb (10 campos).
cliente	Datos maestros de un cliente.	id, nombreCompleto, DNI, direccion, telefono1, telefono2, email, parcela1-3, superficie, observaciones (13

campos).

3. storage.py – Persistencia en archivos de texto

Este módulo actúa como una base de datos muy liviana: cada registro se guarda como una línea en un archivo .txt dentro de la carpeta data/. Contiene utilidades de directorio, generación de IDs, funciones de guardado/carga y tablas de impuestos.

Función	Descripción resumida	Notas clave / advertencias
ensure_data_directory()	Crea (si falta) la carpeta data/ junto al código y devuelve su ruta.	
get_next_cobro_id(), get_next_pago_id(), get_next_clients_id()	Abren el archivo correspondiente y devuelven número de registros + 1 para evitar colisiones.	Si el archivo aún no existe devuelven 1.
save_cobros(cobros_tuple)	Añade cada objeto cobro al final de data/cobros.txt como tupla de 21 campos.	
save_pagos(pagos_tuple)	Añade cada objeto pago al final de data/pagos.txt como tupla de 10 campos.	
save_clients(clients_tuple)	Añade cada objeto cliente al final de data/clientes.txt.	
load_plan_cuentas() / save_plan_cuentas()	Lee y escribe el plan de cuentas contables (código → descripción).	
load_tax_cobros() / save_tax_cobros()	Lee y escribe porcentajes de IIBB y DByCR aplicables por cuenta para cobros.	
load_tax_pagos() / save_tax_pagos()	Lee y escribe porcentaje de DByCR aplicable por cuenta para pagos.	
read_records(path)	Devuelve todas las líneas de un archivo parseadas a tuplas (ast.literal_eval).	
overwrite_records(path, lista)	Sobrescribe por completo el archivo con las tuplas entregadas.	Se usa al eliminar filas en la interfaz.

4. cli.py – Interfaz gráfica (Tkinter)

cli.py construye la ventana principal, los formularios y las tablas. Utiliza ttk (Tkinter moderno) y se apoya en storage.py para guardar/leer datos y en model.py para instanciar objetos.

4.1 Flujo general

• Se importan módulos estándar, Tkinter, y las funciones/clases de model y storage.
• build_styles() define tipografías y tamaños para todos los widgets.
• Utilidades genéricas: read_records(), overwrite_records(), filter_rows().
• class App hereda de Tk, crea la ventana, carga estilos, lee los datos maestros y arma un menú lateral (Cobros, Pagos, Clientes, Listados, Plan de cuentas, Tablas impositivas).
• Cada pantalla se construye bajo demanda. Al guardar cambios se actualiza el archivo y se refresca la lista.

4.2 Elementos principales

_build_list(): Crea listados genéricos (Cobros, Pagos o Clientes) con filtros en tiempo real y botón para eliminar.

_build_cobro() / _save_cobro(): Formulario para ingresos. Calcula subtotal de imputaciones, aplica porcentajes de IIBB y DByCR (consultando load_tax_cobros) y añade IVA 21 %. Guarda un objeto cobro y refresca el listado.

_build_pago() / _save_pago(): Formulario para egresos. Calcula DByCR (% por cuenta) y IVA 21 % sobre el monto neto, crea un objeto pago y lo guarda.

_build_cliente() / _save_cliente(): Alta de nuevos clientes con teléfonos, email y parcelas.

_build_plan(): Gestión del plan de cuentas. Añade o elimina códigos y nombres.

_build_tax_cobros(): Gestión de tabla de impuestos para cobros (IIBB y DByCR por cuenta).

_build_tax_pagos(): Gestión de tabla de impuestos para pagos (solo DByCR por cuenta).