



Documentación del Sistema Contable

Autores: Sandra Julieth Castro Herrera, Diego Armando Giraldo

Fecha: 12 de marzo de 2025

1. Introducción

El sistema contable está diseñado para ayudar a emprendedores y empresas a gestionar sus finanzas de manera eficiente. Permite registrar transacciones, calcular impuestos, generar reportes financieros y administrar cuentas personales y empresariales. A través de la Programación Orientada a Objetos (POO), se estructura en distintas clases que representan entidades clave en el ámbito contable.

2. Arquitectura y UML

El sistema se basa en una arquitectura orientada a objetos, reflejada en el siguiente diagrama UML.

2.1 Clases principales:

Cuenta: Representa una cuenta bancaria con saldo y transacciones asociadas.

Transaccion: Registra ingresos y gastos vinculados a una cuenta.

Reporte: Genera reportes financieros con información de ingresos, gastos y balances.

CuentaPersonal y CuentaEmpresarial: Subclases de Cuenta con funcionalidades específicas.

3. Clases y Atributos

Clase Cuenta

Atributos:

+ titular: String

- idCuenta: String

- saldo: Float

Métodos principales:

+ obtenerBalance(): float

+ agregarTransaccion(t: Transaccion): void

+ eliminarTransaccion(t: Transaccion): void

+ calcularTotalIngresos(): float

+ calcularTotalGastos(): float

+ buscarTransaccion(id: String): Transaccion

Clase Transaccion

Atributos:

+ cuenta: Cuenta

+ idTransaccion: String

+ categoria: String

+ metodoPago: MetodoPago

- monto: float

- tipo: TipoTransaccion

- fecha: Date

- descripcion: String

Métodos principales:

+ obtenerDetalles(): String

+ exportarDatos(formato: String): boolean

+ esIngreso(): boolean

Clase Reporte

Atributos:

+ tipoReporte: TipoReporte

+ ingresos: float

+ gastos: float

+ saldoFinal: float

- fechaInicio: Date

- fechaFin: Date

Métodos principales:

+ generarReporte(): String

+ exportarPDF(): boolean

+ exportarCSV(): boolean

Clases específicas de Reportes

ReporteMensual → + generarResumenMensual(): String

ReporteAnual → + generarResumenAnual(): String

ReporteBalanceGeneral → + generarBalanceGeneral(): String

Clase CuentaPersonal

Atributos:

+ nombreEmprendimiento: String

- ingresosTotales: float

- tasaImpuesto: float

Métodos principales:

+ calcularImpuestos(): float

+ establecerPresupuesto(monto: float): void

+ calcularSaldoDisponible(): float

+ alertarExcesoGastos(): boolean

Clase CuentaEmpresarial

Atributos:

+ nombreEmpresa: String

- ingresosAnuales: float

- gastosAnuales: float

- tasaIVA: float

- salariosEmpleados: List<float>

Métodos principales:

+ generarEstadoFinanciero(): String

+ calcularIVA(): float

+ calcularNomina(): float

+ generarBalanceGeneral(): String

+ calcularDeducciones(): float

4. Getters y Setters

Clase Cuenta

+ getIdCuenta(): String → Retorna el ID de la cuenta (sin setter).

+ getSaldo(): float → Retorna el saldo actual.

+ setSaldo(saldo: float): void → No permite valores negativos.

Clase Transaccion

+ getMonto(): float / + setMonto(monto: float): void → No permite valores negativos.

+ getTipo(): TipoTransaccion / + setTipo(tipo: TipoTransaccion): void → Solo permite INGRESO o GASTO.

+ getFecha(): Date / + setFecha(fecha: Date): void → No permite fechas futuras.

+ getDescripcion(): String / + setDescripcion(descripcion: String): void → Permite modificar.

Clase Reporte

+ getFechaInicio(): Date / + setFechaInicio(fecha: Date): void → Debe ser una fecha válida.

+ getFechaFin(): Date / + setFechaFin(fecha: Date): void → No puede ser anterior a fechaInicio.

Clase CuentaPersonal

+ getIngresosTotales(): float / + setIngresosTotales(ingresos: float): void → No permite negativos.

+ getTasaImpuesto(): float / + setTasaImpuesto(tasa: float): void → Debe estar entre 0% y 100%.

Clase CuentaEmpresarial

+ getIngresosAnuales(): float / + setIngresosAnuales(ingresos: float): void → No permite negativos.

+ getGastosAnuales(): float / + setGastosAnuales(gastos: float): void → No permite negativos.

+ getTasaIVA(): float / + setTasaIVA(tasa: float): void → Entre 0% y 100%.

+ getSalariosEmpleados(): List<float> / + setSalariosEmpleados(salarios: List<float>): void → Modifica la lista.