

Entrega de Actividad - Logica Comutacional

Grupo: 8

Proyecto: 8

Integrantes

- Jeronimo Vaca Vallejo
- Miguel Angel Colmenares Ortiz
- Daniel Donis Rivas Lozano

Asignatura: Logica Comutacional

Horario: Martes y Jueves, de 9:00 a 11:00 am.

a. Presuma el ciclo de vida de construcción de un programa.

El ciclo de vida de un programa sigue varias etapas fundamentales para asegurar su correcto desarrollo y funcionamiento. ¿Cuáles son?

1. Análisis del problema

Antes de empezar a escribir el código, primero hay que entender muy bien el problema.

- Para qué se necesita el programa?

- ¿Qué se debe hacer?

- ¿Quiénes lo van a usar y cómo lo van a usar?

Si por ejemplo se trata de programar un sistema para manejar los pedidos en un restaurante, primero se debe entender cómo funciona el restaurante, es decir, cómo toman los pedidos, cómo se organizan en la cocina, cómo se entrega la comida, etc.

2. Diseño de la solución

Aquí es donde se empieza a planear cómo será el programa.

- Se decide qué partes tendrá el código y cómo se relacionarán entre sí.

- Se hacen diagramas para representar cómo va a funcionar.

Continuando con el ejemplo del restaurante, en esta parte se realizaría un diagrama que muestre cómo se registran los pedidos, cómo se procesan en la cocina y cómo se entregan al cliente.

3. Implementación (construcción del Programa)

Se empieza a escribir el código en un lenguaje de programación.

- Se crean las funciones y clases necesarias.

- Se prueba cada parte del código por separado para asegurarse que funciona bien.

- Se empiezan a ensamblar todas las piezas del programa.

Continuando con el restaurante, esta sería la parte en donde se programan las pantallas donde se ingresan los pedidos, las bases de datos donde

Se guarda la información, y los cálculos para el total de la cuenta.

4. Pruebas y depuración.

Una vez que el programa está construido, hay que asegurarse de que funciona correctamente.

- Se hacen pruebas con distintos datos de entrada.
- Se verifican los cálculos y las respuestas del sistema.
- Se corrigen los errores que puedan suceder.

En el ejemplo del restaurante, en esta etapa, se podrían realizar pruebas como por ejemplo que pasaría si se pide más comida de la disponible o si el mesero olvida ingresar la mesa en el sistema.

5. Entrega y mantenimiento

Cuando se han pasado las etapas anteriores y el programa funciona correctamente, este se entrega a los clientes o los usuarios finales.

- Se les explica cómo usarlo.
- Se deja una documentación con las instrucciones.
- Si con el tiempo hay problemas o se necesitan mejoras, se hacen actualizaciones.

Si por ejemplo en el restaurante se empiezan a vender nuevos platillos, el programa debe actualizarse para incluirlos.

A medida que el programa se usa, se pueden encontrar errores o surgir nuevas necesidades, por lo que veces se tendrá que volver a algunas etapas iniciales para hacer las mejoras.

- **PREFERENCIA EN EL LIBRO:** Página 6, Fig 1.3.

b. Explique los aspectos que hacen parte del análisis de un problema.

Antes de empezar a programar, hay que entender correctamente el problema a resolver. Para ello se puede dividir el análisis en tres partes:

1. Requerimientos funcionales

¿Qué debe hacer el programa?

Son las funciones principales del sistema, es decir, lo que el usuario podrá hacer con él. Por ejemplo:

- En un cajero automático, los requerimientos funcionales serían: retirar dinero, consultar saldo, transferir dinero.
- En un video juego, los requerimientos funcionales, pueden ser mover el personaje, saltar, disparar, guardar la partida.

Al no identificar bien los requerimientos funcionales, se podría terminar haciendo un programa que no resuelve lo que se necesita.

2. Mundo del problema

¿Dónde y cómo se usará este programa?

Usando el ejemplo del cajero automático, el mundo del problema incluye:

- Las cuentas bancarias de los clientes
- La cantidad de dinero disponible en el cajero
- La seguridad del sistema para evitar fraudes.

Si no se entiende bien el mundo del problema, se podría hacer un sistema de cajero automático que permita retirar más dinero del disponible.

3. Requerimientos no funcionales.

Son las condiciones que el programa debe cumplir pero que no tienen que ver directamente con sus funciones.

Por ejemplo

- **Tiempo de respuesta:** Un cajero debe procesar una transacción en pocos segundos

- **Seguridad:** Un sistema bancario debe encriptar la información del usuario.

Compatibilidad: Por ejemplo, un videojuego debe funcionar en distintas consolas o sistemas operativos.

Al no tener en cuenta los requerimientos funcionales, se podría hacer un programa que funcione bien, pero que sea muy tonto, inseguro o difícil de usar.

- **Preferencia en el libro:** Página 5, figura 1.2.

C. Explique las etapas del proceso de solución de problemas.

El proceso de solución de problemas en programación tiene cinco etapas:

1. Análisis del problema

- Se identifican los requerimientos funcionales y no funcionales.
- Se investiga cómo funciona el mundo del problema.

2. Diseño de la solución

- Se decide como se organizará la información y las funciones.
- Se hacen diagramas y se plantea la estructura del código.

3. Implementación

- Se escribe el código en un lenguaje de programación.
- Se prueba cada parte del programa por separado.

4. Pruebas y depuración

- Se verifica que el programa haga lo que se espera.
- Se corrigen errores y se mejora el rendimiento.

5. Entrega y mantenimiento

- Se instala el programa en su entorno final.
- Se hacen mejoras y actualizaciones con el paso del tiempo.

Si se siguen correctamente estas etapas, se podrían evitar costosos errores y nos aseguramos de que nuestro programa sea útil y confiable.

- **Preferencia en el libro:** Página 6, figura 1.3

d. ¿Cuáles son los elementos que se deben entregar a un cliente?

Al terminar de programar no solo se entrega el código. También se deben incluir otros elementos para que el cliente o usuario pueda usarlo correctamente y mantenerlo en el futuro.

Elementos que deben entregarse

1. **Código fuente:** Los archivos del código programado.
2. **Programa ejecutable:** Es la versión lista para usarse sin la necesidad de modificar el código.
3. **Manual de Usuario:** Explicación de cómo usar el programa.
4. **Documentación técnica:** Diagramas y explicaciones sobre cómo funciona el código.
5. **Pruebas y demostraciones:** Son reportes que demuestran que el sistema funciona correctamente.

El propósito de estos elementos es que el cliente pueda usar y actualizar el programa sin problemas en el futuro.

e. Elabore la tarea No. 9 (pág. 5 del texto guía), con el objetivo de identificar los aspectos que forma parte de un problema.

	Nombre	Dibujar el triángulo
	Resumen	El programa debe permitir visualizar un triángulo en pantalla basado en las coordenadas de sus tres puntos
Requerimiento funcional #	Entradas	Coordenada X e Y del primer punto (x_1, y_1) Coordenada X e Y del segundo punto (x_2, y_2) Coordenada X e Y del tercer punto (x_3, y_3) Color de las líneas del triángulo (RGB) Color de relleno del triángulo (RGB)
	Resultado	se dibujar un triángulo en la interfaz gráfica con los colores y coordenadas ingresadas

	Nombre	Cálculo del Perímetro
	Resumen	El programa debe calcular y mostrar la Suma de las longitudes de los tres lados del triángulo.
	Entradas	Coordenadas $(x_1, y_1), (x_2, y_2), (x_3, y_3)$
Requerimiento funcional 2	Proceso	<p>se calcula la distancia entre cada par de puntos usando la fórmula euclídea:</p> $\text{lado} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$ <p>se suman los tres lados</p>
	Resultado	se muestra el valor del Perímetro en la interfaz gráfica.
	Nombre	Cálculo del área
	Resumen	El programa debe calcular y mostrar el área del triángulo usando la fórmula de Herón.
	Entrada	Coordenadas $(x_1, y_1), (x_2, y_2), (x_3, y_3)$
Requerimiento funcional 3	Proceso	<p>se calcula el semiperímetro</p> $S = \frac{\text{lado}1 + \text{lado}2 + \text{lado}3}{2}$ <p>área:</p> $\text{Área} = \sqrt{S(S - \text{lado}1)(S - \text{lado}2)(S - \text{lado}3)}$
	Resultado	se muestra el área en la interfaz gráfica.

Tarea 2: (F)

Requerimiento funcional 1	Nombre	Registro de usuario.
	Resumen	El sistema debe permitir a los usuarios registrarse proporcionando información personal y creando credenciales de acceso.
	Entradas	Nombre, apellido, correo electrónico, número de identificación, contraseña.
Requerimiento funcional 2	Resultado	Usuario registrado en el sistema con credenciales válidas para iniciar sesión.
	Nombre	Depósito de dinero.
	Resumen	El sistema debe permitir a los usuarios realizar depósitos en su cuenta bancaria dentro del simulador.
Requerimiento funcional 3	Entradas	Número de cuenta, monto a depositar.
	Resultado	El saldo de la cuenta del usuario aumenta según el monto depositado.
	Nombre	Retiro del dinero.
	Resumen	El sistema debe permitir a los usuarios retirar dinero de su cuenta siempre que tengan saldo suficiente.
	Entradas	Número de cuenta, monto a retirar.
	Resultado	El saldo de la cuenta del usuario disminuye según el monto retirado, siempre que haya fondos disponibles.

Tarea 3: (9)

Requerimiento funcional 1	Nombre	Clasificación del triángulo.
	Resumen	El sistema debe determinar el tipo de triángulo según las longitudes de sus lados (equilátero, isósceles o escaleno).
	Entradas	Longitud de los tres lados del triángulo.
Requerimiento funcional 2	Resultado	El sistema muestra si el triángulo es equilátero, isósceles o escaleno.
	Nombre	cálculo del perímetro.
	Resumen	El sistema debe calcular el perímetro del triángulo sumando sus tres lados.
Requerimiento funcional 2	Entradas	Longitud de los tres lados del triángulo.
	Resultado	El sistema muestra el valor del perímetro del triángulo.
	Nombre	cálculo del área.
Requerimiento funcional 3	Resumen	El sistema debe calcular el área del triángulo utilizando la fórmula de Herón.
	Entradas	Longitud de los tres lados del triángulo.
	Resultado	El sistema muestra el área del triángulo calculada con la fórmula de Herón.

Tarea 4: (h)

	Nombre	Descripción
Entidad	Triángulo	Figura geométrica con tres lados y tres ángulos. Puede clasificarse según sus lados o sus ángulos.
Entidad	Lado	Segmento de linea que forma parte del triángulo. Se utilizan sus longitudes para clasificar el triángulo y calcular su perímetro.
Entidad	Ángulo	Espacio entre dos lados del triángulo. Se utilizan para clasificar el triángulo en acutángulo, rectángulo o obtusángulo.

Punto de reflexión: ¿Qué pasa si no identificamos bien las entidades del mundo?

- Si no el modelo del sistema puede ser inexacto, lo que afectaría la funcionalidad del programa. Por ejemplo, si no consideramos correctamente los lados y ángulos, podríamos clasificar erróneamente el triángulo o hacer cálculos incorrectos.

Punto de reflexión: ¿Cómo decidir si se trata efectivamente de una entidad y no sólo de una característica de una entidad ya identificada?

- Debemos preguntarnos si tiene identidad propia y si puede tener atributos o comportamientos específicos dentro del sistema. Por ejemplo, un "lado" es una entidad porque tiene un valor de longitud y es esencial para definir un triángulo, mientras que "color" sería solo un atributo porque no afecta la definición geométrica del triángulo.

Tarea S1 (1)

clase: Cuenta Bancaria

Atributo	valores posibles	Diagrama UML
Número de cuenta.	Número único de 10 dígitos.	Cuenta Bancaria
Saldo.	Valor numérico (positivo o negativo).	Número de cuenta Saldo
Titular.	Nombre del propietario de la cuenta.	Titular

clase: Cuenta Corriente

Atributo	valores posibles	Diagrama UML
Límite de crédito.	Valor numérico (máximo permitido en sobreiro).	cuenta corriente
Comisión.	Porcentaje aplicado por mantenimiento de cuenta.	Límite de crédito comisión
Descubierto permitido.	Sí, no	Descubierto permitido

clase: Cuenta Ahorros

Atributo	valores posibles	Diagrama UML
Tasa de interés.	Porcentaje anual aplicado al saldo.	Cuenta Ahorros
Movimientos permitidos.	Número de retiros permitidos por mes.	Tasa de interés Movimientos permitidos
Monto mínimo.	Valor mínimo requerido para mantener la cuenta activa.	Monto mínimo

clase: CDT

Atributo	valores posibles	Diagrama UML
Número de certificado.	Número único de identificación (10 dígitos).	CDT
Monto.	Valor numérico (mínimo establecido por el banco).	Número de certificado Monto
Tiempo.	30, 60, 90, 180, 360 o más.	Tiempo

Clsse: Mes

Atributo	valores posibles	Diagrama UML
Nombre del mes.	Enero, febrero, Marzo, Abril ... Diciembre.	Mes
Día del mes.	1, 2, 3, etc.	NOMBRE DEL MES. DÍA DEL MES.
Es bisiesto	Sí/No.	ES BISIESTO.

Tarea 6: (1)

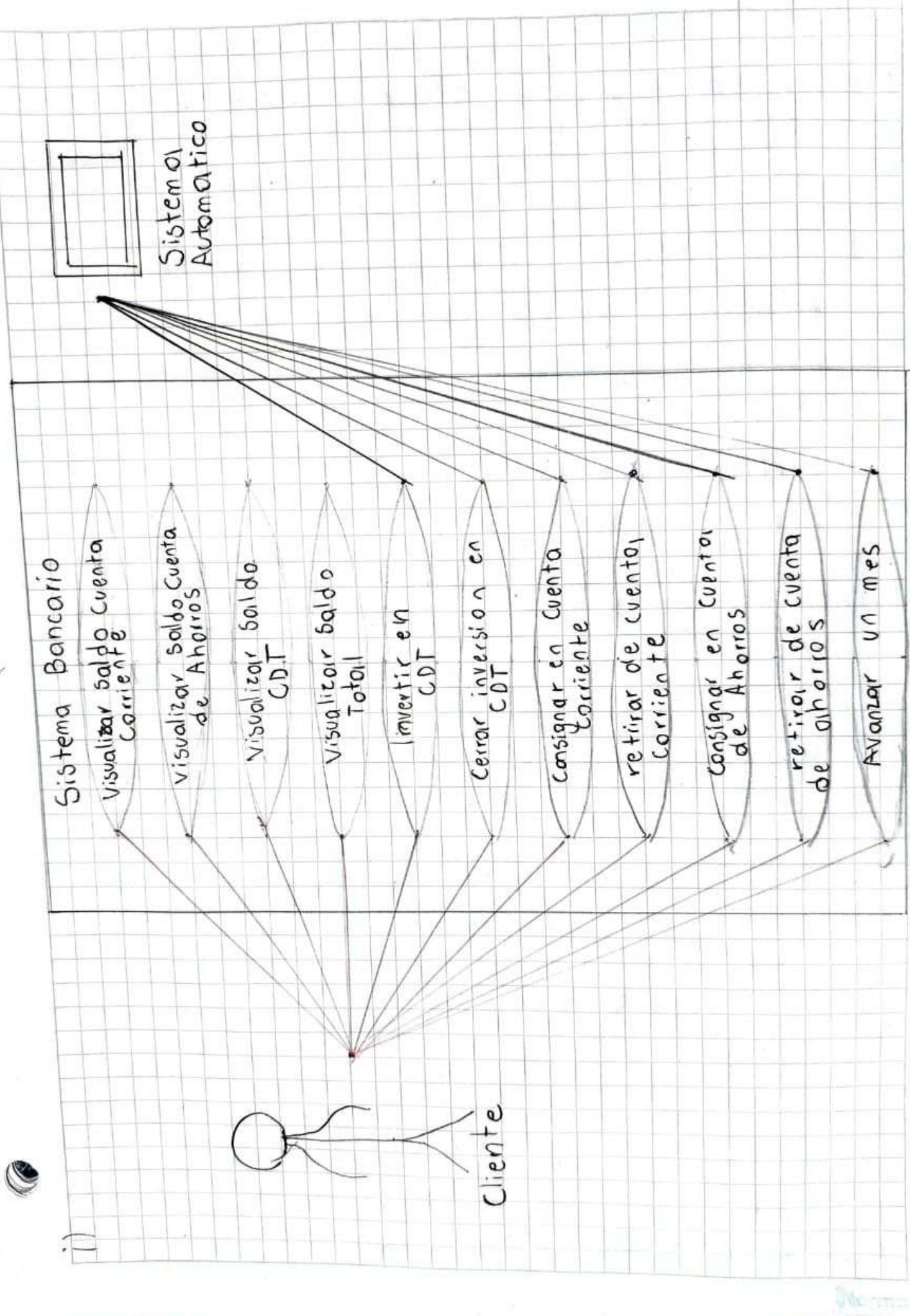
Si las instrucciones no especifican desde qué estación se inicia el recorrido, ni cuál es la estación de destino, el usuario no podrá determinar el camino correcto, además si no se dice la linea del metro el usuario podría equivocarse.

Si el algoritmo dice "Tome la linea correcta", pero el metro va en ambos sentidos podría haber confusión, también si dice "bajar en la tercera estación", podría ser confuso sería mejor si se dijera el nombre de las estaciones.

Por otro lado si suponemos que todos ya conocen el sistema pues obviamente a los turistas sería difícil, además si va alguien que no conozca el idioma francés podría complicarse mucho.

También instrucciones como "Tome la linea azul" podrían no funcionar si hay más líneas azules.

Para que funcione debería ser muy clara y precisa, muy completa y universal para que todo el mundo pueda usarlo sería útil como ya mencioné antes poner el nombre de las estaciones, además de subir la específica con su dirección, número exacto de paradas y tal vez usar un color único por ruta aunque esta última no sería tan práctica.



K.) Control de Gastos Telefónicos Empresariales

- En una empresa con múltiples líneas telefónicas, llevar un control eficiente de los gastos en llamadas es fundamental para la gestión financiera. Actualmente, la empresa cuenta con tres líneas telefónicas, a través de las cuales se pueden realizar llamadas locales, de larga distancia y a celulares, cada una con tarifas específicas:
 - Minuto de llamada local: \$ 35 pesos
 - Minuto de llamada de larga distancia: \$ 380 pesos
 - Minuto de llamada a celular: \$ 990 pesos

El sistema debe permitir registrar llamadas en cada línea, visualizar información detallada de cada una de ellas y obtener un resumen consolidado de los costos y duración de las llamadas. Además, la aplicación debe proporcionar la opción de restablecer la información de las líneas telefónicas cuando sea necesario.

Requerimientos Funcionales

1. Agregar una llamada.

- Entrada: Número de la línea telefónica, tipo de llamada y duración en minutos.
- Salida: Se actualiza el registro de llamadas de la línea correspondiente con la nueva información.

2. Visualizar la información de una línea telefónica

- Entrada: Número de la línea telefónica
- Salida: Se muestra el número total de llamadas

realizadas, la duración total de llamadas realizadas en minutos y el costo total de las llamadas en pesos, según las tarifas establecidas.

3. Visualizar la información consolidada de todos los líneas.

- Entrada: Ninguna

- Salida: Se muestra el número total de llamadas realizadas, la duración total de todas las llamadas en minutos, el costo total de todas las llamadas en pesos y el cálculo del costo promedio por minuto.

4. Reiniciar la información de los líneas telefónicas

- Entrada:

- Salida: Se restablecen los registros de llamadas y costos de todos los líneas telefónicas.

Clases Identificadas en el Modelo de Interfaz

1. Panel Ahorros (JPanel, ActionListener)

- Atributos: etiquetas, botones, cajos de texto
- Métodos: actualizarSaldoAhorros(), actionPerformed()

2. Panel Corriente //

- Atributos: etiquetas, botones, cajos de texto
- Métodos: actualizarSaldoCorriente(), actionPerformed()

3. Panel CDT //

- Atributos: etiquetas, botones, cajos de texto
- Métodos: actualizarSaldoCDT(), actionPerformed()

4. PanelDatosCliente (JPanel)

- Atributos: etiquetaNombre, etiquetacedula, txtNombre + txtCedula.
- Métodos: actualizarNombre(), actualizarCedula()

5. Panel Opciones //

- Métodos: actionPerformed()

Clases Identificadas en el Modelo de Pruebas

1. SimuladorBancarioTest

- Atributos: cuenta: SimuladorBancario
- Métodos: setupEscenario(), testConsignar(), testRetirar, testSaldoTotal, etc

2 CuentaAhorroTest

- Atributos: cuenta: CuentaAhorros

m.)

Clases Identificadas en el Modelo Conceptual

1. Simulador Bancario

- Atributos : cédula, nombre, mesActual
- Métodos : calcularSaldoTotal(), invertirCDT(), consignarCuentaCorriente(), retirarCuentaCorriente(), consignarCuentaAhorros(), retirarCuentaAhorros(), avanzarMesSimulación(), cerrarCDT()

2. CuentaCorriente

- Atributos : Saldo
- Métodos : consignarMonto(), retirarMonto()

3. CuentaAhorros

- Atributos : Saldo , interesMensual.
- Métodos : consignarMonto(), retirarMonto(), actualizarSaldoPorPaisOmes()

4. CDT

- Atributos : valorInvertido, interesMensual, mesApertura
- Métodos : invertir(), calcularValorPresente(), cerrar()

5. InterfazSimulador (6UI)

- Métodos : actualizarCliente(), invertirCDT(), cerrarCDT(), retirarAhorros(), consignarAhorros(), retirarCorriente(), consignarCorriente(), avanzarMesSimulación()

- Métodos : testConsignar(), testRetirar(),
testActualizarDatos()

3. CuentaCorriente Test

- Atributos: cuenta: CuentaCorriente
- Métodos : testConsignar() testRetirar

4 CDT Test

- Atributos: cdt : CDT
- Métodos : testInversion(), testCierre()

n.)

Idea: 1: Control de Gastos Telefónicos

	Nombre	Registro de Llamadas
Requerimiento Funcional	Resumen	Permite registrar cada llamada realizada desde una de las tres líneas telefónicas de la empresa especificando la duración y el tipo de llamada.
	Entradas	Número de línea, tipo de llamada y duración en minutos
	Resultado	Registro de llamada con su costo calculado y asignado a la línea que corresponde
Requerimiento Funcional	Nombre	Consulta de Información por línea
	Resumen	Hace un resumen detallado de las llamadas realizadas por cada línea
	Entradas	Número de línea a consultar
Requerimiento Funcional	Resultado	Información de la línea, como el número total de llamadas, su duración y su costo total
	Nombre	Resumen de Gastos Telefónicos
	Resumen	Genera un informe con la información total de gastos de las 3 líneas
Requerimiento Funcional	Entradas	No tiene (ya que la información ya están)
	Resultado	Total de llamadas, total de minutos y el costo total de cada línea
	Nombre	Reinicio de Datos
Requerimiento Funcional	Resumen	Permite restablecer la información de todas las líneas
	Entradas	Confirmación del usuario para poder hacer el reinicio

Resultado | Todos los datos eliminados dejando todo en 0

Idea 2: Optimización de Gastos telefónicos

requerimiento funcional 1	Nombre	Comparación de tarifas
	Resumen	Analiza el consumo telefónico, y sugiere si hay una tarifa más económica
	Entradas	Historial de llamadas con duración y tipo
requerimiento funcional 2	Resulado	Recomendación de planes alternativos con proyección de ahorro
	Nombre	Notificaciones de consumo
	Resumen	Envía alertas cuando una línea está cerca de un límite de gasto predefinido
requerimiento funcional 3	Entradas	Límite de gasto de la empresa, consumo actual de cada línea
	Resulado	Notificación en Pantalla cuando alcanza el límite
	Nombre	Predicción de costos futuros
requerimiento funcional 4	Resumen	Utiliza datos históricos para predecir gastos
	Entradas	Historial de consumo de llamadas
	Resulado	Estimación de costos futuros basado en tendencias de uso
requerimiento funcional 4	Nombre	Generación de reportes
	Resumen	Hace reportes detallados
	Entradas	Periodo de análisis, y datos
	Resulado	Archivo con informe detallados