

Taller Guía Proyecto 1.

Arcenegas Vargas María Paula
Carollo Núñez Nicoll Stefanny
Rodríguez Rodríguez María Fernanda.

Lógica Computacional (presencial)

Profesor: Juan Carlos Martínez Díaz

Universidad Antonio Narino
Sede Sur
Facultad Ing. Sistemas

Bogotá D.C. - Bogotá D.C.
21 de Septiembre 2024.

Actividades Preliminares.

Contexto.

la especificación de un problema

1. Resuma el ciclo de vida de construcción de un programa
- 2) El ciclo de vida de construcción de un programa, también conocido como ciclo de vida del desarrollo del software (SDLC), abarca varias etapas fundamentales que guían el proceso desde la concepción hasta el crecimiento del programa. A continuación se describen las fases principales:

1. Planificación: En esta fase se establecen los objetivos del proyecto, se identifican los recursos necesarios y se elabora un plan de desarrollo (detallado) que incluye el alcance cronograma y estimaciones de costos.

2. Análisis: Se recopilan y evalúan los requisitos del sistema mediante la interacción con los usuarios finales y otras partes interesadas. Esta fase es crucial para entender las necesidades y expectativas, y se documentan los requisitos funcionales y no funcionales.

3. Diseño: Se desarrolla la arquitectura del sistema basada en los requisitos recopilados. Esto incluye la creación de diagramas de flujo, modelos de datos y decisiones sobre la tecnología a utilizar.

4. Desarrollo: En esta etapa se construyen los componentes del software. Se realiza la programación según las especificaciones definidas en las fases anteriores, asegurando que cada componente funcione correctamente.

5. Pruebas: Se llevan a cabo pruebas exhaustivas para verificar que el software cumple con los requisitos establecidos. Esto incluye pruebas unitarias, de integración y de usuario para detectar y corregir errores antes del lanzamiento.

6. Implementación: El software se despliega en el entorno productivo. Esta fase puede incluir la capacitación de usuarios y la migración de datos, asegurando que todos los aspectos técnicos estén listos para su uso.

7. Mantenimiento: Una vez implementado, el software entra en una fase de mantenimiento donde se corrigen errores, se realizan mejoras y se adaptan cambios según las necesidades de los usuarios. Esta fase puede extenderse durante un período significativo.

8. Cierre: Finalmente, se evalúan los beneficios obtenidos del programa y se documentan las lecciones aprendidas. Esto incluye liberar recursos y finalizar cualquier trabajo pendiente.

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

R) Esto implica comprender la naturaleza que permite entender y resolver problemas, mediante el uso de algoritmos y programación. Los aspectos clave para este análisis son:

1. **Definición del problema:** El primer paso en el análisis es definir claramente el problema; esto implica comprender la naturaleza del problema y lo que se espera lograr, para esto es fundamental establecer un enunciado coherente que describa la situación a resolver, así como identificar los objetivos específicos que se desean alcanzar.

2. **Especificación de entradas y salidas:** Una vez definido el problema, se deben especificar claramente las entradas (los datos disponibles) y las salidas (los resultados esperados), esto incluye:

- **Datos de entrada:** Identificar qué información se tiene disponible para resolver el problema
- **Datos de salida:** Determinar qué información se desea obtener como resultado de la solución del problema.

3. **Análisis del flujo:** Este aspecto implica estudiar cómo se van a manipular los datos de entrada para generar las salidas deseadas. Es crucial entender las interrelaciones entre las entradas y salidas, así como los procesos necesarios para transformar una en otra.

4. **Diseño de algoritmo:** Con base en el análisis previo, se procede a diseñar un algoritmo que describa la secuencia de pasos necesarios para resolver el problema. Un algoritmo debe ser:

- **Pasos:** Debe indicar claramente cada paso a seguir
- **Definido:** Al seguirlo, debe producir el mismo resultado cada vez.
- **Finito:** Debe ser o tener un número limitado de pasos

5. **Codificación:** Una vez diseñado el algoritmo, se convierte en un programa utilizando un lenguaje de programación adecuado. Este proceso implica traducir la lógica del algoritmo código que pueda ser ejecutado por un computador.

6. **Ejecución y validación:** Finalmente, se ejecuta el programa y se valida su funcionamiento. Esto incluye pruebas para asegurarse de que el programa produce las salidas correctas para las entradas dadas y cumple con los objetivos establecidos al principio.

7. **Resolución y mejoría:** El análisis de problemas no es un proceso lineal, es decir, puede requerir iteraciones donde se revisan y ajustan tanto el algoritmo como el programa en función de los resultados obtenidos durante la validación.

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

R) Se compone de varias etapas fundamentales para este proceso:

1. **Definición del problema:** La primera etapa consiste en definir claramente el problema, esto, indica que se tiene que formular un enunciado que describa la situación que se desea resolver y los objetivos que se quieren alcanzar. Es fundamental entender qué se espera obtener al final del proceso para poder avanzar adecuadamente.

2. Análisis de los datos: En esta etapa, se realiza un análisis exhaustivo de los datos necesarios para resolver el problema, esto incluye:

- Identificar las entradas deseables
- Determinar las salidas deseadas
- Evaluar las herramientas y métodos que se pueden utilizar para manipular los datos y alcanzar la solución

3. Diseño de la solución: Una vez que el problema está definido y los datos analizados, se procede a diseñar una solución, así:

- Desarrollar un algoritmo que describa los pasos necesarios para resolver el problema.
- Representar el algoritmo mediante diagramas de flujo o en un lenguaje natural, lo cual facilita su comprensión y posterior codificación

4. Codificación: La siguiente etapa es la codificación, donde el algoritmo diseñado se traduce a un lenguaje de programación específico. Esto incluye:

- Escribir el código fuente que será comprendible para el programador
- Compilar el código para convertirlo en un programa ejecutable que pueda ser entendido por la máquina

5. Prueba y depuración: Una vez se ha codificado el programa, es esencial realizar pruebas para verificar su funcionamiento, así:

- Ejecutar el programa con diferentes conjuntos de datos de entrada.
- Identificar y corregir errores o fallas (depuración) para asegurar que el programa produzca las salidas correctas.

6. Documentación: La documentación es una etapa crucial que a menudo se pasa por alto. consiste en registrar:

- La descripción del problema y la solución implementada.
- Detalles sobre cómo usar el programa y cualquier consideración importante sobre su funcionamiento.

Esto facilita la comprensión futura del código y su mantenimiento.

7. Mantenimiento: Finalmente, después de la implementación, es necesario realizar un mantenimiento continuo del software. Esto incluye:

- Actualizaciones para mejorar funcionalidades o corregir errores.
- Adaptaciones a cambios en los requisitos o en el entorno operativo.

Este proceso iterativo asegura que la solución sea efectiva y relevante a lo largo del tiempo.

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

La entrega de un proyecto o solución a un cliente en el ámbito de la computación implica varios elementos esenciales que garantizan que el cliente recibe un producto completo y funcional. Aquellos elementos son:

1. Documentación del proyecto: Esta es crucial y debe incluir

- **Requerimientos del sistema:** Un documento que detalle las necesidades y expectativas del cliente, así como las especificaciones funcionales y no funcionales del sistema.
- **Manual de usuario:** Instrucciones claras sobre cómo utilizar el sistema, incluyendo ejemplos y guías paso a paso.
- **Documentación técnica:** Información sobre la arquitectura del sistema, diseño de bases de datos, diagramas de flujo, y detalles sobre la implementación técnica.

2. Código fuente: El código fuente del software desarrollado debe ser entregado al cliente, esto incluye:

- **Archivos de código:** Todos los archivos necesarios para ejecutar el software.
- **Comentarios y estructura:** Un código bien comentado que facilite su comprensión y mantenimiento en el futuro.

3. Pruebas y resultados: Es importante proporcionar evidencia de que el software funciona como se esperaba:

- **Resultados de prueba:** Informes que muestran los resultados de las pruebas realizadas, incluyendo pruebas unitarias, de integración y funcionales.
- **Registro de errores:** Documentación sobre cualquier error encontrado durante las pruebas y cómo se resolvieron.

4. Capacitación y soporte: Para asegurar una correcta (implementación) implementación, se deben ofrecer:

- **Sesiones de capacitación:** Formación para los usuarios finales sobre cómo utilizar el sistema.
- **Soporte técnico:** Información sobre cómo acceder al soporte técnico en caso de problemas futuros.

5. Plan de mantenimiento: Un plan que detalle cómo se manejarán futuras actualizaciones y mantenimiento esencial del sistema:

- **Mantenimiento preventivo:** Estrategias para garantizar que el sistema funcione correctamente a lo largo del tiempo.
- **Actualizaciones futuras:** Información sobre cómo implementarán futuras mejoras o cambios en el sistema.

6. Acuerdos contractuales: Finalmente, es fundamental incluir cualquier acuerdo contractual relacionado con la entrega:

- **Términos y condiciones:** Acuerdos que establecen las obligaciones de ambas partes, incluyendo tiempos de entrega, costos, y penalizaciones por incumplimiento.

Acción

Enunciado

Se quiere crear un Programa que permita simular el comportamiento de las cuentas bancarias de un cliente. Un cliente puede tener tres productos financieros básicos:

1. Una Cuenta corriente. El cliente puede depositar o retirar dinero. Pero no recibe ningún interés por el dinero que se encuentra allí depositado.
2. Una cuenta de ahorros. El cliente puede depositar o retirar dinero y recibe un interés mensual del 0.6% sobre el saldo actual.
3. Un Certificado de depósito a término CDT. Cuando el cliente abre un CDT, define la cantidad de dinero que quiere invertir y negocia con el banco el interés mensual que va a recibir. A diferencia de la cuenta Corriente o la cuenta de ahorros, en un CDT no se puede consignar ni retirar dinero. La única operación posible es cerrarlo, en cuyo caso el dinero y sus intereses pasan a la cuenta corriente.

Estos productos son independientes y tienen comportamientos particulares. El saldo total de la cuenta es la suma de lo que el cliente tiene en cada uno de dichos productos.

El Programa debe permitir al usuario:

- 1 Visualizar el saldo de la cuenta corriente del cliente.
- 2 Visualizar el saldo de la cuenta de ahorros del cliente.
- 3 Visualizar el saldo del CDT del cliente.
- 4 Visualizar el saldo total que tiene el cliente en los productos del banco.
- 5 Invertir un monto de dinero en un CDT.
- 6 Cerrar la inversión en CDT.
- 7 Consignar un monto de dinero en la cuenta corriente del cliente.
- 8 Retirar un monto de dinero en la cuenta de ahorros del cliente.
- 9 Consignar un monto de dinero en la cuenta de ahorros del cliente.
- 10 Retirar un monto de dinero en la cuenta de ahorros del cliente.
- 11 Avanzar en un mes la simulación.

Requerimientos Funcionales

Nombre= R1- Visualizar el saldo de la cuenta corriente del Cliente.

Nombre= R2- Visualizar el saldo de la cuenta de ahorros del Cliente.

Nombre= R3 Visualizar el saldo del CDT del cliente

Nombre= R4- Visualizar el saldo total que tiene el cliente en los productos del banco.

Nombre= R5- Intervenir un monto de dinero en un CDT.

Nombre = R6 - Cerrar la Inversión en CDT.

Nombre = R7 - Consignar un monto de dinero en la cuenta corriente del cliente.

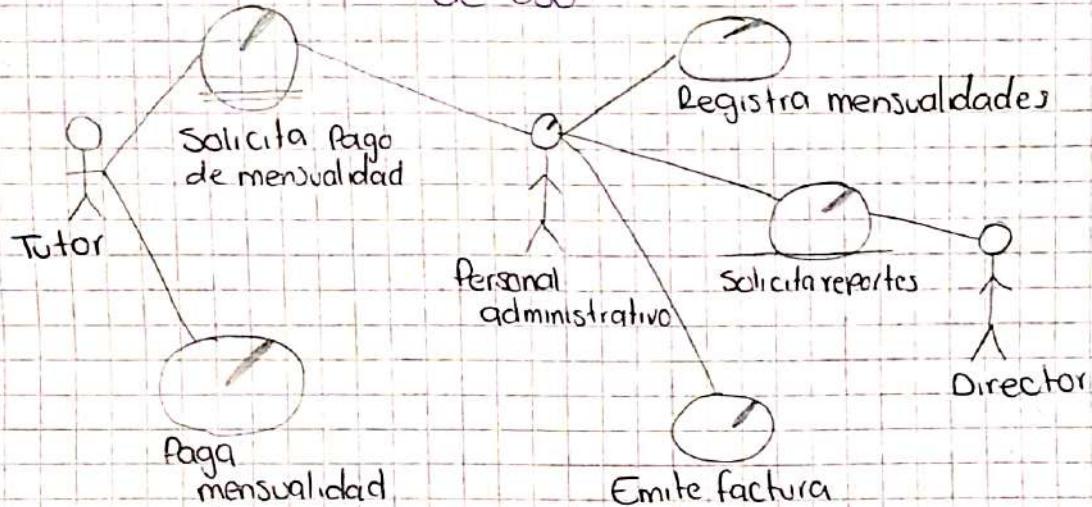
Nombre = R8 - Retirar un monto de dinero de la cuenta corriente del cliente.

Nombre = R9 - Consignar un monto de dinero en la cuenta de ahorros del cliente.

Nombre = R10 - Retirar de la cuenta de ahorros

Nombre = R11 - Avanzar un mes la simulación bancaria.

diagrama de casos de uso



modelo conceptual

Simulador Bancario

```
- Cedula: string  
- Nombre: string  
- mesActual: int  
  
+ SimuladorBancario(Cedula: string, pNombre: string)  
+ darNombre(): string  
+ darCedula(): string  
+ darCuentaCorriente(): CuentaCorriente  
+ darCuentaAhorros(): CuentaAhorros  
+ darCDT(): CDT  
+ darMesActual(): int  
+ CalcularSaldoTotal(): double  
+ IntervenirCDT(pMonto: double, pInteresMensual: double): void  
+ ConsignarCuentaCorriente(pMonto: double): void  
+ ConsignarCuentaAhorros(pMonto: double): void  
+ RetirarCuentaCorriente(pMonto: double): void  
+ RetirarCuentaAhorros(pMonto: double): void  
+ avanzarMesSimulacion(): void  
+ CerrarCDT(): void  
+ metodo1(): String  
+ metodo2(): String
```

CDT

```
- ValorInvertido: double  
- InteresMensual: double  
- MesApertura: int
```

```
+ CDT()
```

```
+ darInteresMensual(): double  
+ invertir(pMontoInvertido: double, pInteresMensual: double,  
          pMes: int): void  
+ CalcularValorPresente(pMesActual: int): double  
+ Cerrar(pMesActual: int): double
```

Cuenta Corriente

```
- Saldo: double
```

```
+ CuentaCorriente()  
+ Invertir(pMontoInvertido: double, pInteresMensual: double,  
          pMesActual: int): double  
+ ConsignarMonto(pMonto: double): void  
+ RetirarMonto(pMonto: double): void
```

Cuenta Ahorros

```
- Saldo: double  
- InteresMensual: double
```

```
+ CuentaAhorros()  
+ darSaldo(): double  
+ darInteresMensual(): double  
+ ConsignarMonto(pMonto: double): void  
+ RetirarMonto(pMonto: double): void  
+ ActualizarSaldoPorPasoMes(): void
```

E1 Elabore la tarea 1 (pgs) con el objetivo de identificar los aspectos que forman parte de un problema

El problema: Un banco quiere crear un programa para manejar sus cajeros automáticos, dicho programa debe permitir retirar dinero y consultar el saldo de los mismos.

Cliente	Banco (Entidad Financiera que requiere el Software)
Usuario	Clientes del banco que utilizan los cajeros automáticos
Requerimiento Funcional	Permitir a los usuarios retirar dinero y consultar el saldo de su cuenta
Mundo del Problema	Cajeros automáticos que deben ser gestionados por el software, incluyendo la seguridad y la disponibilidad del servicio
Requerimiento no funcional	El sistema debe ser seguro, fácil de usar, y debe funcionar 24/7 sin interrupciones

F1 Elabore la tarea 2 (Pg 13) con el objetivo de identificar los requerimientos fundamentales del problema

Problema: Crear habilidad en la identificación y especificación de requerimientos fundamentales

Requerimiento Funcional 1	
Nombre	Registro del Usuario
Resumen	Permitir a los usuarios crear una cuenta proporcionando información personal y de contacto.
Entradas	Nombre, Apellido, Correo Electronico, Contraseña, Telefono.
Resultados	Cuenta creada exitosamente y Usuario Autenticado

Requerimiento Funcional 2	
Nombre	Transferencia de fondos
Resumen	Permitir a los Usuarios Transferir dinero entre cuentas dentro del Simulador.
Entradas	Cuenta de Origen, Cuenta de destino, Monto a Transferir.
Resultados	Transfencia completada y Saldo actualizado en ambas Cuentas.

Requerimiento Funcional 3	
Nombre	Consulta de Saldo
Resumen	Permitir a los Usuarios consultar el Saldo de su cuenta en cualquier momento.
Entradas	Número de cuenta. Autenticación del Usuario
Resultado	Muestra el Saldo actual de la cuenta

5) Elabore la req 3 (pag 141) con el objetivo de identificar los requerimientos (funcionales del problema)

Problema = para el caso de estudio 3. Un programa para manejar un triangulo. Identifique y explique 3 requerimientos (funcionales)

Requerimiento fundamental 1	
Nombre	calcular el area del triangulo
Resumen	Este requerimiento permite al usuario calcular el area de un triangulo dado a su base y altura
Entradas	base (float) altura (float)
resultado	area (float)

Requerimiento fundamental 2	
Nombre	determinar el tipo de triangulo
Resumen	Este requisito identifica el tipo de (equilatero, isosceles, escaleno) basado en la longitudo de sus lados
Entradas	lado 1 (float) lado2 (float) lado 3 (float)
resultado	Tipo de triangulo (String)

Requerimiento fundamental 3	
Nombre	calcular perimetro del triangulo
Resumen	Este requisito permite al usuario calcular el perimetro de un triangulo sumando las longitudes de sus lados
Entradas	lado 1 (float) lado 2 (float) lado 3 (float)
resultado	perimetro (float)

H) Elabore la tarea N⁴ (pag 17) con el objetivo de identificar las entidades del mundo del problema

Identificar el entorno del caso y identificar las entidades del problema identificar las entidades del mundo para el caso de estudio de un programa que maneje un triángulo

Entidad	Nombre	Descripción
1	Triángulo	Una figura geométrica de 3 lados y 3 ángulos
2	Lado	Cada una de las tres líneas que forman el triángulo
3	Ángulo	La medida de la apertura entre 2 lados del triángulo

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

Si no identificamos correctamente las entidades, podríamos tener un malentendido de problema lo que lleva a soluciones incorrectas o ineficaces.

¿Cómo decidir si se trata efectivamente de una entidad y no solo de una característica de otra entidad?

Para decidir si algo es una entidad, debemos evaluar si tiene una existencia independiente y si puede ser descrito de manera única (en lugar de ser simplemente una propiedad o característica de otra entidad).

II) Elabore la tarea 5 (pag 20) con el objetivo de identificar las entidades de un banco de estudio

Problema: Para cada uno de los 5 entidades en el caso de estudio de simulación bancario, identifique los atributos, sus valores posibles, y escriba la clase en UML, no incluyan las relaciones que podrían existir entre las clases. Ya que eso se hará en la siguiente etapa de análisis.

Clase	Atributo	Diagrama de UML
Cuenta	Balance	
Atributo	Valores Posibles	
Número de cuenta	123456789, 987654321	
Saldo	0, 1000, 2500, 5000	
Tipo de cuenta	Ahorros, corriente, CDT	
Fecha de apertura	2023-01-01, 2023-06-15	

- cuenta corriente

Atributo	Valores posibles	Diagrama UML
Número de cuenta	123456789, 987654321	
Saldo	0, 500, 1500, 3000	
Límite		
Sobregiro	0, 500, 1000	
fecha de apertura	2023-01-01, 2023-06-15	
- cuenta de ahorros		

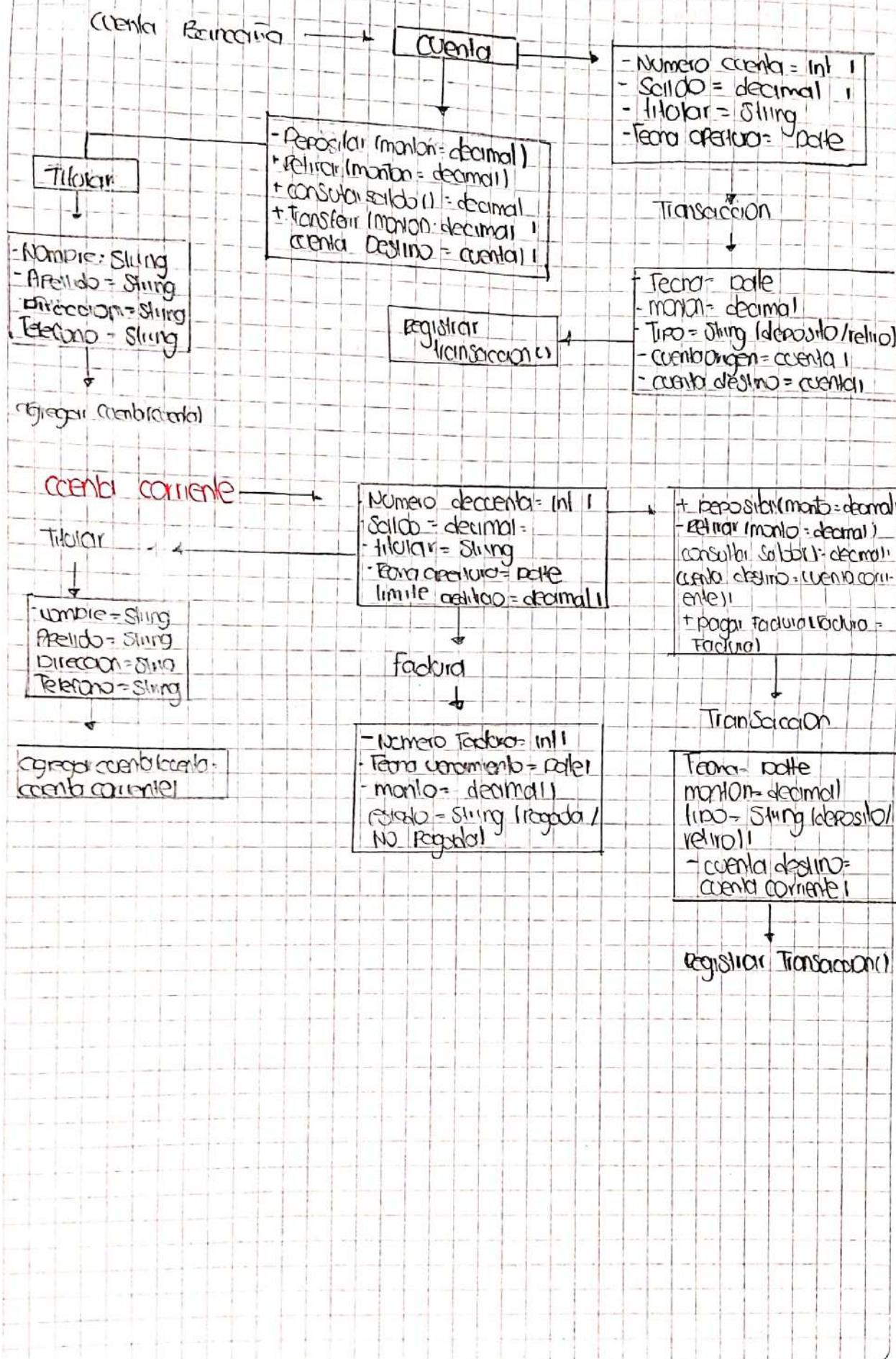
Atributo	Valores posibles	Diagrama UML
Número de cuenta	123456789, 987654321	
Saldo	0, 100, 1000, 2000	
Tasa de interés	15 %, 20 %, 25 %	
fecha de apertura	2023-01-01, 2023-06-15	
clase	CPT	

Atributo	Valores posibles	Diagrama UML
Número de CDT	123456789, 987654321	
Monto	1000, 5000, 100000	
Tasa de interés	30 %, 35 %, 40 %	
Plazo	30 días, 60 días, 180 días	

clase de mes

Atributo	Valores posibles	Diagrama UML
Nombre	Enero, Febrero, Marzo, Abril, Mayo, Junio, Julio, Agosto, Septiembre, Octubre, Noviembre, Diciembre	
Número	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12	
Días	28, 29, 30, 31	
Estación	Invierno, Primavera, Verano, Otoño	

Diagramas de clases (UML)



Clases XML para CDT

Inversor

Nombre = String
Apellido = String
direccion = String
telefono = String

Transaccion

Fecha = Date
monto = decimal
Tipo = String (deposito /
retiro)
CDT = CDT

Registro Transaccion()

Nombre CDT = int

Monto = decimal

Plazo = int

tasa Interes = decimal

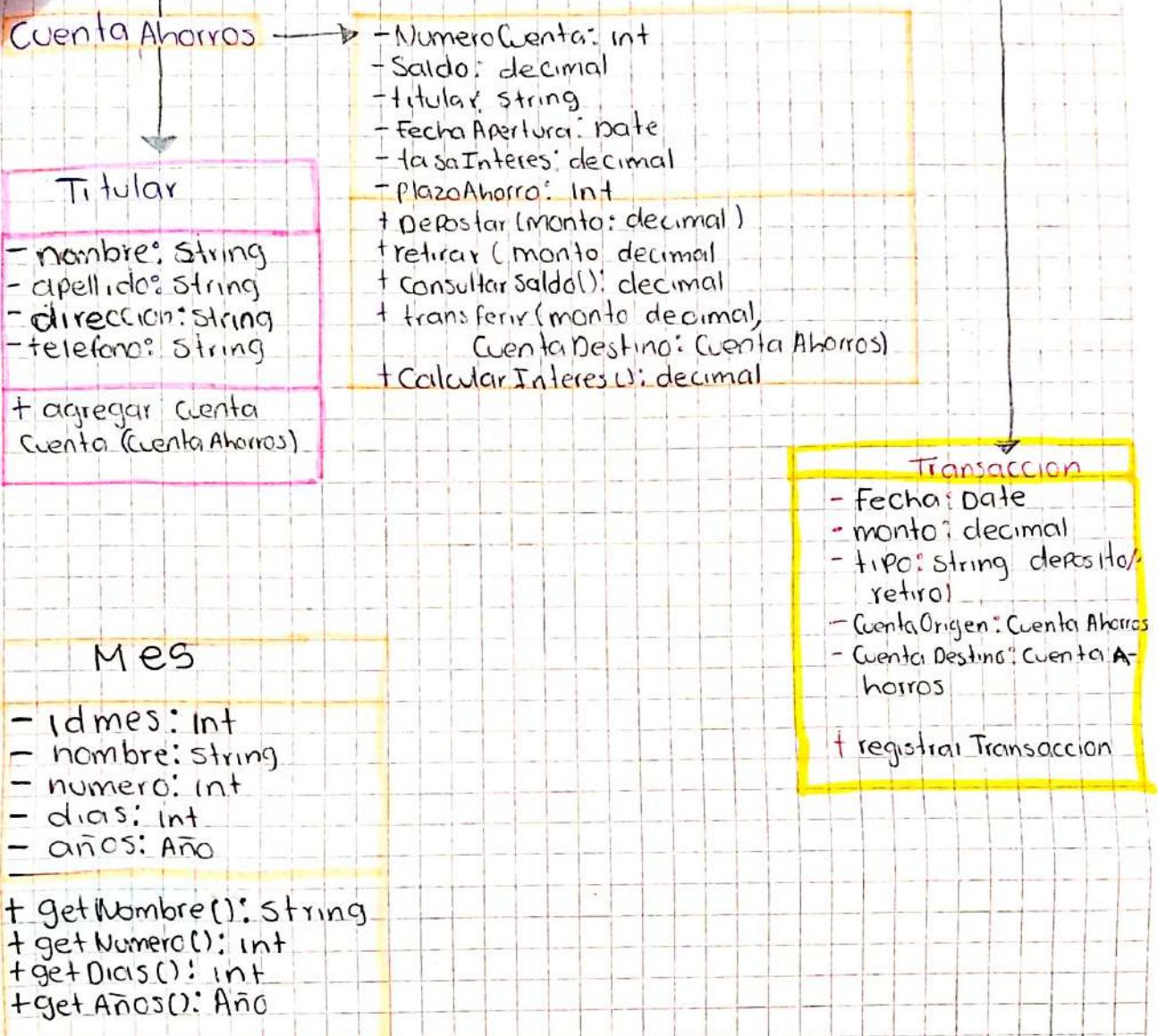
Fecha Vencimiento = Date

Estado = String (adivio / vencido)

crear CDT (monto, Plazo, tasa
interes) +

consultar Saldo () = decimal

consultar = decimal



II) Elabore la idea 6 (Pg. 23) con el objetivo de reflexionar sobre el nivel de precisión de un algoritmo

- Supongamos que usted es la persona que va a circular el logotipo anterior para moverse en el metro de París. Identifique qué problemas podría tener con las instrucciones anteriores

Problemas potenciales:

Interpretación Ambigua de Instrucciones:

Pueden ser interpretadas de diferentes maneras. Por ejemplo, si se indica "tome la siguiente salida", puede no estar claro si se refiere a la proxima salida en la estación o a la siguiente salida en el camino.

Variaciones en el "sentido común":

Se asume que todos los usuarios tienen el mismo nivel de conocimiento. Sin embargo, un turista no puede entender referencias locales de estaciones.

Diferencias culturales:

Las personas de diferentes culturas pueden tener distintas formas de interpretar direcciones. Por ejemplo, en algunos países, "a la derecha" puede ser interpretado de manera diferente dependiendo de la orientación inicial.

Problemas de comunicación:

Si las instrucciones no son claras o están mal redactadas, los usuarios pueden confundirse y tomar decisiones incorrectas.

Se proponen varias medidas, las cuales de utilidad (anotaciones claras, pruebas de usuarios con diversos grupos, mantener la información adicional y agregar información visual como mapas).

Ident 1: Sistema de monitoreo de salud Animal

Requerimiento Funcional 1	Nombre	Monitoreo de salud Animal
	Resumen	Desarrollar un sistema que permita monitorear la salud de mascotas a través de un dispositivo portátil que registre datos vitales como la temperatura, frecuencia cardíaca y Actividad física.
	Entradas	Datos de salud (temperatura, frecuencia cardíaca, Actividad física).
	Resultados	Informe diario de la salud del animal.
Requerimiento Funcional 2	Nombre	Monitoreo de salud Animal
	Resumen	Desarrollar un sistema que permita monitorear la salud de las mascotas a través de un dispositivo portátil que registre datos vitales como la temperatura, frecuencia cardíaca y Actividad física.
	Entradas	Alertas de salud (valores fuera de rango).
	Resultados	Notificación al dueño a través de una aplicación móvil.
Requerimiento Funcional 3	Nombre	Monitoreo de salud Animal
	Resumen	Desarrollar un sistema que permita monitorear la salud de las mascotas a través de un dispositivo portátil que registre datos vitales como la temperatura, frecuencia cardíaca y Actividad física.
	Entradas	Historial médico del Animal
	Resultados	Recomendaciones de cuidado y visitas al veterinario
Requerimiento Funcional 4	Nombre	Monitoreo de salud Animal
	Resumen	Desarrollar un sistema que permita monitorear la salud de las mascotas a través de un dispositivo portátil que registre datos vitales como la temperatura, frecuencia cardíaca y Actividad física.

U
A
O
E
calidad

Entradas	Datos de alimentación y ejercicio.
Resultados	Plan de dieta y ejercicio Personalizado

Idea proyecto 2: Sistema de monitoreo y gestión de calidad del agua.

Requerimiento Funcional 1	Nombre	Aquadafe
	Resumen	Crear un sistema inteligente para monitorear la calidad del agua en tiempo real en ríos, lagos y embalses. Utilizando sensores IoT (Internet de las cosas) el sistema puede medir parámetros como la contaminación, el pH, la temperatura y el oxígeno disuelto. Los datos recogidos se utilizan para alertar a las autoridades y a la comunidad sobre la salud del ecosistema acuático.
	Entradas	Parámetros de calidad del agua (pH, turbidez, oxígeno disuelto)
	Resultado	Informe en tiempo real de la calidad del agua.
Requerimiento Funcional 2	Nombre	Aquadafe
	Resumen	Crear un sistema inteligente para monitorear la calidad del agua en tiempo real en ríos, lagos y embalses. Utilizando sensores IoT (Internet de las cosas) el sistema puede medir parámetros como la contaminación, el pH, la temperatura y el oxígeno disuelto. Los datos recogidos se utilizan para alertar a las autoridades y a la comunidad sobre la salud del ecosistema acuático.
Requerimiento Funcional 3	Entradas	Datos históricos de calidad del agua
	Resultado	Ánalisis de tendencias y predicciones sobre la calidad del agua.
Requerimiento Funcional 3	Nombre	Aqua Safe
	Resumen	Crear un sistema inteligente para monitorear la calidad del agua en tiempo real en ríos, lagos y embalses. Utilizando sensores IoT (Internet de las cosas) el sistema puede medir parámetros como la contaminación, el pH, la temperatura

		Y el oxígeno disuelto. Los datos recogidos se utilizan para alertar a las autoridades y a la comunidad sobre la salud del ecosistema acuático.
	Entradas	Información sobre fuentes de contaminación (industrial y agrícola)
	Resultado	Mapa interactivo que muestra las fuentes de contaminación y su impacto en el ecosistema.
Requerimiento funcional 4	Nombre	Aqua Data
	Resumen	Crear un sistema inteligente para monitorear la calidad del agua en tiempo real en ríos, lagos y embalses. Utilizando sensores IoT (Internet de las cosas) el sistema puede medir parámetros como la contaminación, el pH, la temperatura, y el oxígeno disuelto. Los datos recogidos se utilizan para alertar a las autoridades y a la comunidad sobre la salud del ecosistema acuático.
	Entradas	Alertas de datos críticos (niveles de contaminación elevados).
	Resultados	Notificaciones automáticas a las autoridades locales y a la comunidad.
Requerimiento funcional 5	Nombre	Aquadata
	Resumen	Crear un sistema inteligente para monitorear la calidad del agua en tiempo real en ríos, lagos y embalses. Utilizando sensores IoT (Internet de las cosas) el sistema puede medir parámetros como la contaminación, el pH, la temperatura y el oxígeno disuelto. Los datos recogidos se utilizan para alertar a las autoridades y a la comunidad sobre la salud del ecosistema acuático.
	Entradas	Opiniones y reportes de la comunidad sobre el estado del agua.
	Resultados	Plataforma de participación ciudadana que permite a la comunidad contribuir a la vigilancia del agua.