

Actividad

1. Resuma el ciclo de vida de un programa.

El ciclo de vida de un programa se define por varias etapas que sigue el desarrollo del software, desde el inicio de su implementación y mantenimiento que incluye las siguientes etapas:

- Requisitos y planificación: se recogen y se analizan las necesidades del cliente y se crea para establecer los objetivos del proyecto.
- Diseño del software: se crea una arquitectura y diseño detallado del programa para determinar los componentes, estructura del código y sus interacciones.
- Codificación e implementación: describe el código fuente del programa siguiendo el diseño previamente establecido.
- Mantenimiento: que implementa las correcciones de errores, mejoras o adaptaciones a nuevos requisitos.

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

En el análisis de un problema es un proceso crucial en la implementación de un software, esto nos permite descomponer el problema para entender mejor y encontrar la solución adecuada.

Para ello es entender esencialmente que el lugar se tiene que resolver, implica identificar los síntomas y sus causas, en esto también implica las necesidades del cliente, usuario y el sistema esto se enfoca en los requisitos, funcionales y no funcionales.

Esto nos permite dividir problemas en partes más pequeñas y manejables y limitando los problemas.

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

- Es importante identificar del problema esto nos permite identificar y entender claramente la naturaleza del problema, sus síntomas y su alcance. Es de manera precisa para evitar y buscar una solución.
- Esto nos ayuda analizar un problema para poder entender su causa, recopilar información (relevante) relevante, factores y sus problemas, para poder ir a los síntomas, efectos.

Tarea 1.

- En este punto nos indica que cuando tengo el problema identificado nos permite identificar y plantear unas ideas exactas para implementar en el código para generar la corrección del error.

- Cuando tengo las alternativas, es importante evaluar en cada una y considerar factores para seleccionar la mejor solución, adaptando el código y ofreciendo las mejores soluciones.

- Cuando implementamos la solución esto nos permitirá realizar unas evaluaciones detalladas si en este caso el problema se soluciona de una manera eficiente y efectiva.

- Nos permite reflexionar sobre todo el proceso para identificar lecciones aprendidas y mejoras que se puede aplicar con el prototipo.

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

Para entregar un producto o proyecto para el cliente este proporciona una lista de elementos importantes que aseguran que el producto cumpla con las expectativas y sea gestionado de forma correcta.

1. Producto (software o prototipo web)
2. Documentación técnica (manual del usuario, documentación técnica, requisitos del sistema)
3. Código fuente (si aplica)
4. Licencias y derechos de uso
5. Instrucciones de instalación o despliegue.
6. Plan de mantenimiento y soporte.
7. Resultados de los pruebas (si aplica)
8. Acta de aceptación del cliente
9. Facturación y acuerdos finoscieros
10. Capacitación (si aplica)

Tarea 1.

Objetivo: Identificar los aspectos que forman parte de un problema

- El problema: Un banco quiere crear un programa para manejar sus cajeros automáticos. Dicho programa solo debe permitir retirar dinero y consultar el saldo de una cuenta.

Identifique y discute los aspectos que constituyen el problema. Si el enunciado no es explícito con respecto a algún punto, intente imaginar la manera de completarlo.

Elemento	Descripción
Cliente	Banco o entidad financiera que desea implementar el programa para sus cajeros automáticos.
Usuarios	Clientes del banco que utilizan los cajeros automáticos para retirar el dinero o consultar el saldo
Requerimiento funcional	<ol style="list-style-type: none">1. Permitir el retiro del dinero2. Consultar el saldo de la cuenta del cliente
mundo del Problema	<ul style="list-style-type: none">- El programa debe estar conectado a los cuentos bancarios del cliente y actualizar en tiempo real- Se deben seguir protocolos de seguridad bancaria y autenticación para proteger las transacciones
Requerimientos no funcionales	<ol style="list-style-type: none">1. Seguridad: implementación de cifrado y autenticación segura (por ejemplo: PIN).2. Tiempo de respuesta: el sistema debe ser rápido y eficiente, con una respuesta en tiempo real3. Disponibilidad: el sistema debe estar disponible 24/7.4. Compatibilidad: el software debe ser compatible con hardware de cajeros automáticos existentes5. Interfaz intuitiva: la interfaz debe ser fácil de usar.

Tarea 2.

- **Objetivo:** Crear habilidad en la identificación y especificación de requerimientos funcionales para el caso estudio 2, un simulador bancario, identifique y especifique tres requerimientos funcionales.

Nombre	Resumen	Entrada	Resultado
Simulación de intereses mensuales	El sistema debe calcular y aplicar automáticamente un 0.6% de interés en cuentas de ahorros sobre el saldo de la cuenta de ahorros del cliente.	- Saldo inicial en la cuenta de ahorros (0,6%)	- Saldo actualizado en la cuenta de ahorros.
Simulación de ahorro de meses	El programa debe permitir avanzar el tiempo avanzar meses para el cliente ver como evolucionan sus productos financieros, aplicando los intereses y centavos correspondientes.	- Saldo en la cuenta de ahorros.	- Evolución del saldo en el producto.
Operaciones de depósito y retiro en cuentas	El sistema debe permitir al cliente realizar depósitos y retiros en su cuenta corriente y de ahorros, actualizando los saldos de forma inmediata.	- cantidad a consignar o retirar.	- Saldo actualizado tras realizar la operación mostrando el nuevo balance en la cuenta.

Tarea. 3: Crear habilidad en la identificación y especificación de requerimientos funcionales

Para el caso estudio 3, un programa para manejar un Triángulo, identifique y especifique tres requerimientos funcionales.

Nombre	Resumen	Entradas	Resultados
Cambiar los puntos del triángulo.	el programa debe permitir al usuario modificar las coordenadas (x, y) de los tres puntos que definen el triángulo.	nuevas coordenadas (x, y) para los puntos P_1, P_2 y P_3	el triángulo se redibuja con las nuevas coordenadas y se recalcule automáticamente el perímetro, área, y altura.
Calcular y mostrar el área y el Perímetro	el sistema debe calcular el área y perímetro del triángulo con base en las coordenadas de los tres puntos y mostrar los resultados en la Pantalla	coordenadas actuales de los tres puntos del triángulo (P_1, P_2 y P_3)	Área, perímetro y altura del triángulo mostrados en la zona de "medidas en pixeles" actualizados en tiempo real
Cambiar el color del triángulo	el programa debe permitir al usuario cambiar el color del triángulo y los líneas seleccionadas con un color RGB.	Valores RGB (Rojo, Verde, Azul) seleccionados por el usuario para el relleno y las líneas	el color del triángulo cambie en la pantalla aplicando el nuevo color del relleno y de las líneas según los valores RGB ingresados por el usuario

Tarea 4: Identificar las identidades del mundo para el caso estudio 3
un programa que maneje un triángulo

En el enunciado del caso y trate de guiarse por los sustantivos para
identificar las entidades del mundo del problema.

Entidad	Descripción
Triángulo	Figura geométrica definida por 3 puntos en el plano, que se utiliza en el programa, tiene un color de líneas y un color de relleno.
Punto	Cada triángulo está formado por 3 puntos (P_1, P_2, P_3) cada uno de los cuales tiene coordenadas X e Y que determinan su posición en el espacio.
Coordenadas X, Y	Valores numéricos que definen la posición de los puntos en el plano. Cada punto tiene una coordenada X (horizontal) y una coordenada Y (vertical).
Color	Definido por un valor RGB (Red-Green-Blue) utilizado tanto para el color de los líneas como para el color de relleno del triángulo.
RGB	Sistema de representación de colores basados en tres valores entre 0 y 255 que representan las intensidades de rojo (R), verde (G) y azul (B).
Perímetro	longitud total de los lados del triángulo calculada a partir de los distancias entre los tres puntos.
Área	Espacio dentro del triángulo, calculado a partir de las coordenadas de los tres puntos.
Altura	Distancia entre la base del triángulo y el punto más alto calculada con las coordenadas de los puntos del triángulo.
Líneas	Lados del triángulo que conectan los puntos P_1, P_2, P_3 . Cada línea tiene un color que puede ser cambiado por el usuario.
Pantalla	Lugar donde se visualiza el triángulo y sus medidas, como el perímetro, el área y la altura.
Usuario	Persona que interacciona con el programa, modificando los puntos, colores y visualizando el triángulo y sus propiedades geométricas.

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

= Se maneja de la siguiente forma por medio de autorama en el modelo, Relaciones, persistencia y su comportamiento propio.

¿Qué pasa si no identificamos correctamente las entidades del mundo?

Puede presentar lo siguiente: Falta de flexibilidad, problemas en el diseño del modelo de relaciones, Duplicación de información.

Tarea 5: Para cada una de las 4 entidades indentificadas en el caso esbozo del simulador bancario, identifique los atributos, sus valores posibles, escriba la clase en UML, no incluya las relaciones que puedan existir entre las clases, ya que eso lo haremos en la siguiente etapa del análisis. Por ahora trate de identificar las características de las entidades que son importantes para los requerimientos funcionales.

- Cliente.

Atributo	Valores Posibles	Descripción
nombre	Cualquier cadena de caracteres	Nombre completo del cliente
Cédula	Cualquier número entero	Documento del cliente
Edad	Número positivo entre (0-18)	edad del cliente
Tipo de cli.	Regular, Premium, corporativo	Tipo de cliente

- UML Cliente

Cliente
documento: int
nombre: String
edad: int
Tipo-cliente: String

- Cuenta ahorros.

Atributo	Valores posibles	Cuenta ahorros
# cuenta	Cadena alfanumérica	# cuenta: string
Saldo	Número decimal ≥ 0	Saldo: float
Interés mensual	0,6% por defecto	Interés: float
Estado cuenta	Activa, Inactiva, Bloqueado	Estado: String

- Cuenta corriente.

Atributo	Valores Posibles	Cuenta corriente
# cuenta	Cadena alfanumérica	# cuenta: string
Saldo	Número decimal ≥ 0	Saldo: float
Estado cuenta	Activa, Inactiva, Bloqueado	Estado: String

- CDT

Atributo	Valores Posibles	CDT
monto_inversion	Número decimal ≥ 0	Monto_inversion: float
interés_mensual	Número decimal entre 0 a 100%	Interés_mensual: float
Plazo	Número entero positivo ≥ 1	Plazo: int
Estado CDT	Abierto, Cerrado	Estado_CDT: string

- Simulador bancario

Atributos	Valores Posibles	Simulador Bancario
cliente	Objeto cliente	cliente: Cliente
cuenta ahorro	Objeto cuenta ahorro	cuenta ahorro: Cuenta ahorro
cuenta corriente	Objeto cuenta corriente	cuenta corriente: Cuenta corriente
CDT	Objeto CDT	CDT: CDT
mes simulación	numero entero positivo	mes simulación: int
saldo total	numero decimal >= 0	saldo total: float

Tarea 6: Reflexionar sobre el nivel de precisión que debe ser usado en un algoritmo para evitar ambigüedades.

Suponga que usted es la persona que va a utilizar el algoritmo anterior, para moverse en el metro de París. Identifique que problemas podría tener con las instrucciones anteriores, piense por ejemplo si están completas.

El algoritmo que nos presenta para moverse en el metro parisino contiene una secuencia de pasos que aparentemente es lógica, pero si analizamos con más detalle, podemos identificar varios puntos en lo que puede presentar ambigüedades o falta precisión al tratarse de un problema real.

Problemas con las instrucciones.

1. Paso 1: Compra un boleto de vía en los puntos de venta que se encuentran a la entrada de cada una de las estaciones del metro.

- Problema: no se especifica como realizar la compra. ¿Cómo se realiza el pago en efectivo o tarjeta? ¿se compra en una máquina o en una taquilla?

- Ambigüedad: las instrucciones suponen que el personaje entiende como funcionan los puntos de venta y esto familiarizado.

2. Paso 2: Identifica en el mapa del metro la estación donde está y el punto donde ir.

- Problema: ¿Qué sucede si el personaje no encuentra el mapa del metro?, no se da ninguna orientación adicional de como obtener un mapa.

- Ambigüedad: el algoritmo no completa situaciones donde los mapas no se encuentre disponibles o no sea legible.

7.3. Paso 3: Localice el nombre de la estación del metro más cercana.

- Problema: no se determina exactamente cuál es la estación más cercana al destino, para personas que no conozcan la ciudad.

- Ambigüedad: puede presentar varias estaciones cercanas a un destino.

4. Paso 4: Verifique si a partir de donde está, hay una línea que pase por la estación del destino.

- Problema: El algoritmo asume que la persona sabe cómo buscar líneas de metro y entender el mapa, pero no todos los personas entiende.

- Ambigüedad: no se especifica qué hacer si no hay una línea directa hacia la estación del destino.

5. Paso 5: Si encuentra la línea, busque el nombre de la misma en la dirección de destino.

- Problema: no se indica qué hacer si no se encuentra la línea o si hay múltiples opciones, la misma línea puede ir en varias direcciones.

- Ambigüedad: no incluye un criterio claro para elegir la dirección correcta.

6. Paso 6: Sube al metro en el andén de la línea identificada en el paso anterior y bójese en la estación del destino.

- Problema: ¿Qué sucede si el pasajero se pierde?, no cuenta o no explica cómo identificar el orden correcto.

- Ambigüedad: Se asume que la persona sabrá automáticamente cuál es el orden correcto y como identificarlo.

Mejores recomendaciones

- Incluir instrucciones sobre que hacer en caso de que no hay una linea recta, como identificar el lugar del paradero correcto
- Tener una explicación exacta de como realizar el pago, como encontrar un mapa y como hacer transferencia.
- Proveer diferentes versiones del algoritmo dependiendo de si la persona si vive en el metro o si tiene experiencia

14. Estudie el siguiente aspecto del ejemplo seleccionando: enunciado, requerimientos funcionales (caso de uso) y el modelo de clases del proyecto. A continuación, redacta el cronograma del problema y el nombre cada uno de los requerimientos funcionales del proyecto.

- Enunciado del problema: es implementar una plataforma de gestión electoral que nos permita el proceso del registro y consulta de información de los candidatos, votos, y estadísticas, relacionadas con un proceso electoral simulado, la aplicación debe calcular costos de campaña basados en el medio de comunicación que influye en cada voto.

- Requerimientos funcionales: Por medio de este parte nos permite identificar los siguientes requerimientos funcionales, por medio de ello se ve a evidencia de la siguiente forma.

- Registro del candidato
- Registro del voto
- Calcular costo de la campaña
- Calcular el total de los votos
- Calcular costo promedio por campaña
- Generar la estadística por rango de edad y género
- Información del candidato
- Vaciar la urna

- Modelo (clases del proyecto)

- **Candidato**: Representa la información del candidato
- **Votos**: Representa un voto promedio/entido y almacenar la información, solicita el candidato seleccionado
- **Elección**: Presenta el proceso electoral en su conjunto y contiene métodos para gestionar candidatos, votos y estadísticas.

Clase Candidato {

 nombre: String

 edad: Integer

 partido político: String

 costo campaña: Double

 númerovotos: Integer.

}

Clase Voto {

 candidato: candidato

 medio de comunicación: String

}

Clase elecciones {

 candidato: list < candidato >

 votos: list < Votos >

 registrar candidato(candidato)

 registrar voto(votos)

 calcular costo promedio(candidato)

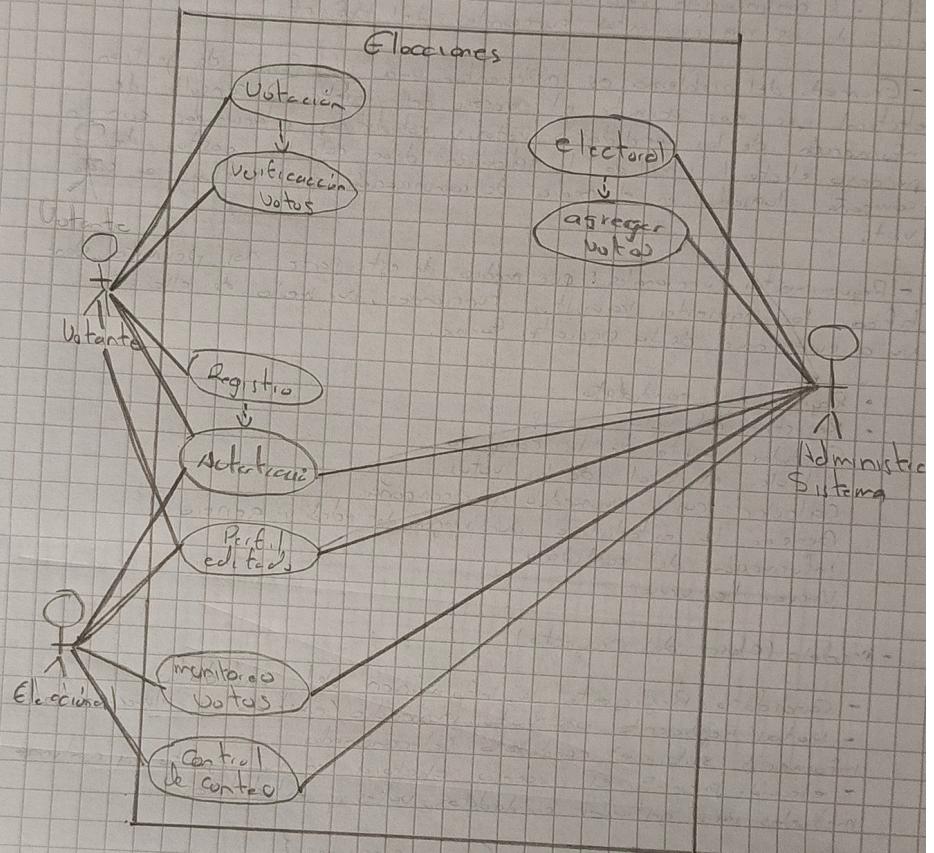
 calcular total votos(): Integer

 calcular costo promedio(): Double

 generar estadísticas(): map<String, Integer>

 vaciar urna()

I. Dibuja el respectivo diagrama de caso de uso del ejemplo elegido
(consulta "requisimientos funcionales" en respectivo 'archivo').



m. Observa nuevamente el modelo conceptual del caso y escribe el nombre de cada una las clases identificando sus respectivas variables (Atributos) y funciones:

Atributos	
Clase Candidate	<ul style="list-style-type: none">• nombre: string• Edad: Integer• Partido político: string• costo_electrónico: Double• num_votos: int• porcentaje_votos: double•
Funciones	
	<ul style="list-style-type: none">• mostrar_información• calcular_porcetaje(total_votos: int)• incrementar_votos(medio: string)• Vaciar_urna():

Atributos	
Clase Elección	<ul style="list-style-type: none">• candidatos: list<candidate>• Total_votos: int
Funciones	
	<ul style="list-style-type: none">• mostrar_información_candidatos(): Votar por candidato (candidato: medio: string) Calcular total de votos (): Calcular_ganador (): restar_votos (porcentaje_votos: int): Vaciar_urna ():

n. debes plantear 2 ideas del proyecto (problema, soluciones y algoritmico)

Problema: el objetivo es desarrollar una aplicación que permita a los usuarios votar por un candidato mientras se calcula automáticamente el costo de compra en función del medio publicitario

Solución: el proyecto puede usar un algoritmo de agrupación y calculo

- Sumar los votos por candidato
- calcular el porcentaje de voto por candidato
- determinar el costo de la compra

Problema 2: Es un objetivo que también genera una estadística dentro gráficas por voto de edad y género.

Solución:

- registrar los datos de votación (edad y género)
- clasificar los votos
- calcular las estadísticas.

Requerimientos funcionales

Requerimiento funcional	Entradas	nombre	Visualización de la información de un candidato.
	Resumen	Entradas	Selección del candidato, candidato 1, candidato 2, candidato 3.
	Resumen	Resumen	La aplicación debe permitir al usuario visualizar la información detallada de los tres candidatos.
1.	Resultados	Resumen	Se despliega en pantalla la información del candidato seleccionado nombre, edades, partidos políticos, costo de compra, número de votos y el porcentaje.
	Entradas	nombre	Votar por un candidato
	Resumen	Resumen	La aplicación debe permitir a los usuarios votar por un candidato, indicando el medio de comunicación que influye su decisión.
Requerimiento funcional	Entradas	Entradas	• Selección del candidato por el que desea votar
			• Selección del medio de comunicación que influye el voto
2.	Resultados		• el voto se suma al total del candidato
			• el costo de la compra del candidato se incrementa según el medio de comunicación
			• el porcentaje de votos se recalcula con respecto al total de votos.

Requerimiento funcional 3.	nombre	Calcular el número total de votos
	Entradas	Ninguna entrada directa por parte del usuario (acceder mediante un botón).
	Resumen	La aplicación debe mostrar el número total de votos emitidos en la elección.
Requerimiento funcional 4.	Resultados	Se muestra en pantalla el número total de votos cuando se suman los votos de las tres candidatas.
	nombre	Calcular el costo promedio por campaña.
	Resumen	La aplicación debe calcular y mostrar el costo promedio de las campañas de los 3 candidatos.
Requerimiento funcional 5.	Entradas	Ninguna acción directa por parte del usuario.
	Resultados	Se muestra en pantalla el costo promedio, calculado como la suma de los costos de campaña de los 3 candidatos dividida por 3.