



UNIMINUTO
Corporación Universitaria Minuto de Dios
Educación de calidad al alcance de todos
Vigilada MinEduación



**FACULTAD DE INGENIERIA
INGENIERIA DE SISTEMAS**

DESARROLLO BASADO EN PLATAFORMAS

ACTIVIDAD I

**AUTORES:
CAMILA MARTINEZ
CRISTIAN NOVOA**

**DOCENTE:
JUAN CARLOS MARTINEZ**

BOGOTÁ D.C

Línea de Atención al Usuario: 593 30 04 • Línea Nacional: 01 8000 936670
www.uniminuto.edu

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

El ciclo de vida de construcción de un programa generalmente sigue estos pasos

1. **Análisis del problema:** Se define el problema que resolverá con el programa
2. **Diseño:** Se planifica la estructura del programa, algoritmos y diagramas de flujo
3. **Codificación:** Se escribe el código en un lenguaje de programación
4. **Pruebas y depuración:** Se verifica que el programa funcione correctamente y se corrigen errores
5. **Implementación:** Se despliega el programa en un entorno real
6. **Mantenimiento y actualización:** Se realizan mejoras y correcciones a lo largo del tiempo

b. Aspectos del análisis de un problema

El análisis de un problema incluye

1. **Identificación del problema:** Se comprende la necesidad o el desafío a resolver
2. **Recolección de requisitos:** Se determinan las necesidades del usuario y las restricciones del sistema
3. **Definición de entradas y salidas:** Se identifican los datos de entrada y los resultados esperados
4. **Limitaciones y restricciones:** Se consideran aspectos como tiempo de ejecución, recursos y seguridad
5. **Alternativas de solución:** Se evalúan diferentes enfoques para resolver el problema

c. Etapas del proceso de solución de problemas

Consta de las siguientes etapas

1. Comprensión del problema: Analizar en detalle el problema y sus requerimientos
2. Planificación de la solución: Diseñar el algoritmo o estrategia para resolver el problema
3. Implementación de la solución: Escribir el código o desarrollar la solución planteada
4. Pruebas y validación: Verificar que la solución funcione correctamente en distintos escenarios
5. Documentación y mantenimiento: Registrar el funcionamiento del sistema y mejorar su rendimiento

d. Elementos a entregar a un cliente

Los elementos que se deben de entregar a un cliente en un proyecto de software incluye

1. Código fuente: Si el cliente requiere acceso al código del programa
2. Manual de usuario: Documento con instrucciones para usar el sistema
3. Manual técnico: Información sobre la arquitectura y el mantenimiento del software
4. Documentación de requisitos y diseño: Especificaciones técnicas y funcionales del sistema
5. Producto final: El software instalado y listo para su uso
6. Capacitación: Formación para los usuarios finales si es necesario
7. Soporte técnico y garantía: Asistencia en caso de errores o mejoras

e. Tarea No. 1

Cliente	El banco que desea implementar el programa para gestionar sus cajeros automáticos
Usuario	Los clientes del banco que utilizarán los cajeros automáticos para realizar transacciones
Requerimiento funcional	<p>R1: El programa debe permitir a los usuarios retirar dinero de sus cuentas</p> <p>R2: El programa debe permitir a los usuarios consultar el saldo de sus cuentas</p>
Mundo del problema	<ul style="list-style-type: none"> • Es necesario conocer la información de las cuentas de los usuarios, como el número de cuenta, el saldo disponible y la identificación del titular • El programa debe tener acceso a la base de datos del banco para verificar la autenticidad de las transacciones y actualizar los saldos después de cada retiro • Debe haber un sistema de seguridad para autenticar a los usuarios, como el uso de tarjetas y PINs • El programa debe manejar posibles errores, como fondos insuficientes o fallos en la conexión con la base de datos
Requerimiento no funcional	<ul style="list-style-type: none"> • Seguridad: El programa debe garantizar la seguridad de las transacciones y la privacidad de la información de los usuarios • Disponibilidad: Los cajeros automáticos deben estar operativos las 24 horas del día, los 7 días de la semana • Usabilidad: La interfaz del cajero debe ser intuitiva y fácil de usar para los clientes • Rendimiento: El programa debe procesar las transacciones de manera rápida y eficiente para evitar largos esperos

F. Tarea 2

1.

Nombre	R1: Realizar un depósito en una cuenta
Resumen	Permite a un usuario depositar una cantidad de dinero en su cuenta bancaria.
Entradas	Número de cuenta Cantidad a depositar
Resultado	El saldo de la cuenta ha sido incrementado con la cantidad depositada.

2.

Nombre	R2: Realizar un retiro de una cuenta
Resumen	Permite a un usuario retirar una cantidad de dinero de su cuenta bancaria.
Entradas	Número de cuenta Cantidad a retirar
Resultado	El saldo de la cuenta ha sido disminuido con la cantidad retirada siempre que haya fondos suficientes.

3.

Nombre	R3: Consultar el saldo de una cuenta
Resumen	Permite a un usuario consultar el saldo actual de su cuenta bancaria.
Entradas	Número de cuenta
Resultado	Se muestra el saldo actual de la cuenta al usuario.

Observaciones

6. Tarea 3

1. Requerimientos

Nombre	R1: Determinar el tipo de triángulo
Resumen	Permite determinar si un triángulo es equilátero, isósceles o escaleno basado en las longitudes de sus lados
Entradas	longitudes de los tres lados del triángulo
Resultado	Se muestra el tipo de triángulo (equilátero, isósceles o escaleno)

2. Requerimiento

Nombre	R2: Calcular el perímetro del triángulo
Resumen	Permite calcular el perímetro de un triángulo sumando las longitudes de sus tres lados
Entradas	longitudes de los 3 lados del triángulo
Resultado	Se muestra el perímetro del triángulo

3. Requerimiento

Nombre	R3: Calcular el área del triángulo
Resumen	Permite calcular el área de un triángulo utilizando la fórmula de Herón
Entradas	longitudes de los 3 lados del triángulo
Resultado	Se muestra el área del triángulo

H. Tarea 4

	Nombre	
Entidad	Triángulo	La entidad principal que representa un triángulo con tres lados y tres ángulos
Entidad	Lado	Representa cada uno de los tres lados del triángulo, con una longitud específica
Entidad	Ángulo	Representa cada uno de los tres ángulos del triángulo con una medida en grados

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

Podemos enfrentar serios desafíos en el diseño y desarrollo del programa. Un mal diseño basado en entidades incorrectas puede resultar una estructura confusa y difícil de mantener, lo que complica la implementación de funcionalidades. Además, podríamos implementar características que no se alinean con las necesidades reales del problema.

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

Es importante considerar varios factores, primero evaluar si el concepto puede existir de manera independiente y si tiene sus propias características y comportamientos. Si es así, es probable que sea una entidad. Segundo, analizar si el concepto tiene relaciones significativas con otras entidades, lo que también sugiere que es una entidad en sí mismo.

I. Tarea 5

Clase: Cuenta Bancaria

Atributo	Valores Posibles
numeroCuenta	String (Identificador de la cuenta)
titular	String (Nombre del titular de la cuenta)
Saldo	Double (Saldo actual de la cuenta)
fechaApertura	String (Fecha de apertura de la cuenta)
Estado	String (Estado de la cuenta: Activa, Inactiva, bloqueada)

• Diagrama UML de Cuenta Bancaria

Cuenta Bancaria
<ul style="list-style-type: none"> - numeroCuenta: String - titular: String - Saldo: Double - FechaApertura: String - estado: String

• Clase: CuentaCorriente

Atributo	Valores Posibles
limiteSobregiro	double
tasaInteres	double

• Diagrama UML Cuenta Corriente

Cuenta Corriente
<ul style="list-style-type: none"> - limiteSobregiro: Double - tasaInteres: double

• Clase: Cuenta Ahorro

Atributo	Valores Posibles
tasaInteresAnual	double (Tasa de interés anual)
saldoMinimo	double (Saldo minimo requerido para cubrir cargos)

• Diagrama UML de Cuenta Ahorro

Cuenta Ahorro
<ul style="list-style-type: none"> - tasaInteresAnual: Double - saldoMinimo: double

• Diagrama UML de CDT

CDT
<ul style="list-style-type: none"> - numeroCDT: String - montoInvertido: double - Plazo: int - tasaInteres: double

• Clase: CDT

Atributo	Valores Posibles
numeroCDT	String (Identificador único del CDT)
montoInvertido	double (Monto invertido en el CDT)
Plazo	int (Plazo en meses del CDT)
TasaInteres	double (Tasa de interés del CDT)

• Diagrama UML de Mes

Mes
<ul style="list-style-type: none"> - numeroMes: int - nombreMes: String

• Clase: Mes

Atributo	Valores Posibles
numeroMes	int (Número del mes en la simulación)
nombreMes	String (Nombre del mes)

J. Tarea 6

Conclusiones :

El algoritmo presentado para viajar en el metro de París, aunque proporciona una guía general, carece de suficiente precisión y detalle para evitar ambigüedades. Por ejemplo, no especifica cómo identificar correctamente las estaciones en el mapa, cómo distinguir entre múltiples líneas que podrían llevar al destino, o qué hacer en caso de transferencias entre líneas. Además, supone que el usuario tiene un conocimiento previo del sistema de metro y utiliza su "sentido común" para interpretar las instrucciones, lo que puede llevar diferentes interpretaciones y errores. Para que un algoritmo sea efectivo, debe ser lo suficientemente detallado y claro para cualquier persona.

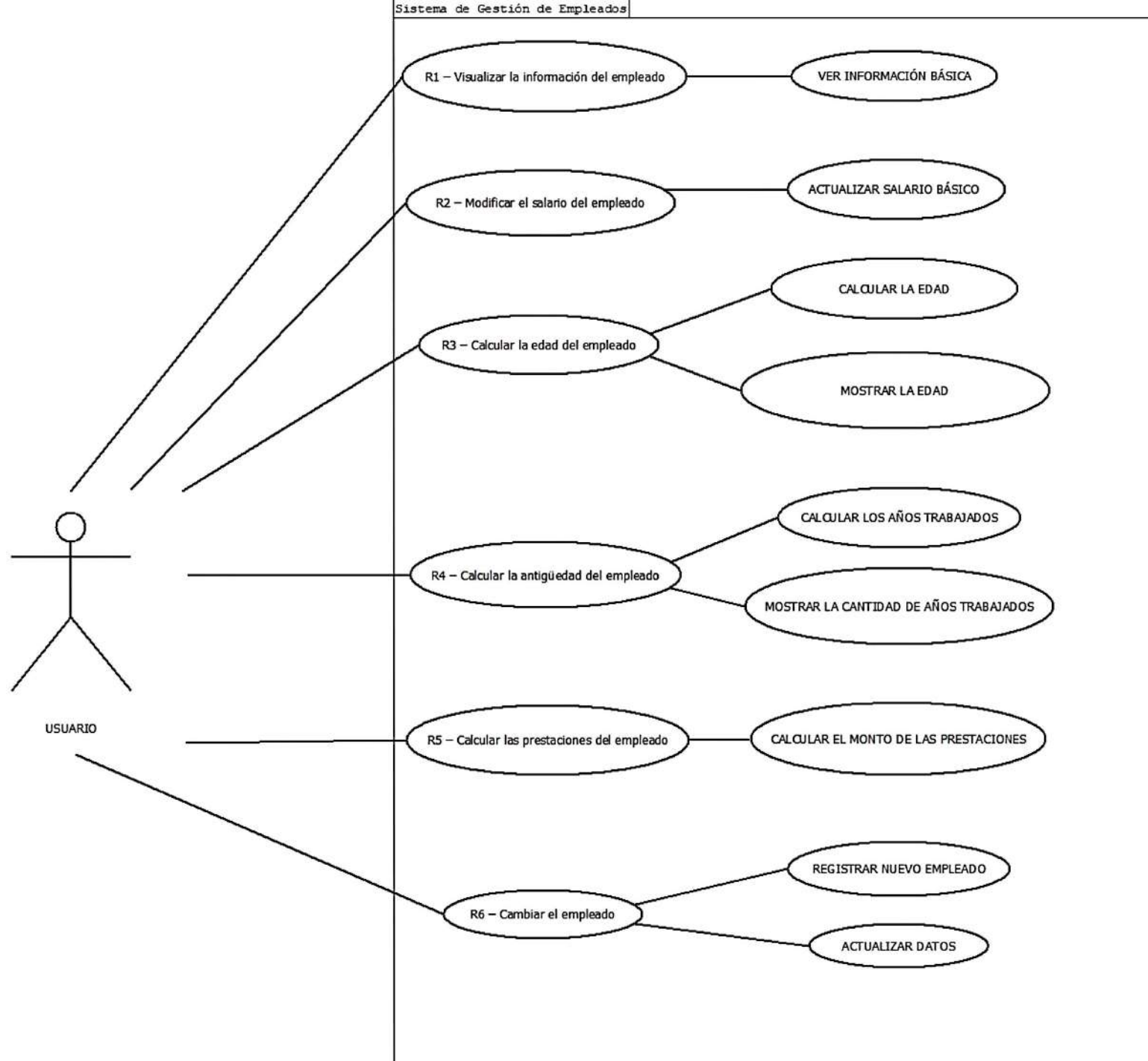
k. Ejemplo 1: El empleado

• Enunciado del problema:

Se requiere una aplicación que permita gestionar la información de un empleado, proporcionando funcionalidades para visualizar sus datos, modificar su salario, calcular la edad, antigüedad y prestaciones, así como cambiar el empleado registrado en el sistema.

• Requerimientos funcionales

1. R1 - Visualizar la información del empleado
2. R2 - Modificar el salario del empleado
3. R3 - Calcular la edad del empleado
4. R4 - Calcular la antigüedad del empleado
5. R5 - Calcular las prestaciones del empleado
6. R6 - Cambiar el empleado



M. Punto m

Clase: Empleado

Atributos (variables)

- nombre (String): Nombre del empleado
- apellido (String): Apellido del empleado
- género (String): Género del empleado
- fecha_nacimiento (Date): Fecha de nacimiento del empleado
- fecha_ingreso (Date): Fecha de ingreso a la empresa
- salario (Double): Salario básico del empleado
- imagen (String): Ruta o URL de la imagen del empleado

N. Punto N

Idea de proyecto 1: Sistema de gestión de tareas y prioridades

Requerimiento Funcional 1	Nombre: RegistrarTarea
	Resumen: Permite al usuario registrar una nueva tarea con su descripción, prioridad y tiempo estimado
	Entradas: Descripción (String), Prioridad (Int), Tiempo (Double)
	Resultado: La tarea se añade a la lista de tareas pendientes
Requerimiento Funcional 2	Nombre: AsignarPrioridad
	Resumen: Permite al usuario asignar o modificar la prioridad de una tarea existente
	Entradas: ID de la tarea (Int), nueva prioridad (Int)
	Resultado: La prioridad de la tarea se actualiza en la lista
Requerimiento Funcional 3	Nombre: CalcularTiempoTotal
	Resumen: Calcula el tiempo total estimado para completar todas las tareas pendientes
	Entradas: Ninguna
	Resultado: Devuelve el tiempo total estimado (Double)

Requerimiento Funcional 4	Nombre :	Optimizar Tareas
	Resumen :	Utiliza un algoritmo para reorganizar las tareas basándose en su prioridad y tiempo estimado
	Entradas:	Ninguna
	Resultado:	La lista de tareas se reordena para optimizar la productividad

Idea de Proyecto: 2 : Sistema de recomendación de Rutas de viaje

Requerimiento Funcional 1	Nombre	Registrar Destino
	Resumen	Permite al usuario registrar un nuevo destino con su nombre, coordenadas y costo asociado
	Entradas	Nombre (String), Coordenadas (double, double), Costo (double)
	Resultado	El destino se añade a la lista de destinos disponibles
Requerimiento Funcional 2	Nombre	Calcular Ruta Óptima
	Resumen	Utiliza un algoritmo (como Dijkstra) para calcular la ruta más eficiente entre dos destinos
	Entradas	Destino Inicial (String), Destino Final (String)
	Resultado	Devuelve la ruta óptima y el costo total
Requerimiento Funcional 3	Nombre	Mostrar Ruta
	Resumen	Muestra la ruta calculada en un formato legible para el usuario
	Entradas	Ruta (List<String>)
	Resultado	La ruta se muestra en la interfaz del usuario
Requerimiento Funcional 4	Nombre	Actualizar Costo
	Resumen	Permite al usuario actualizar el costo asociado a un destino
	Entradas	Nombre del Destino (String), Nuevo Costo (Double)
	Resultado	El costo del destino se actualiza en la lista