GC-F-004v.08

Análisis y desarrollo de software

FASE 3 Ejecución – GA7

Septiembre 16 de 2025

@SENAComunica

www.sena.edu.co

GA7

1. IDENTIFICACIÓN DE LA GUÍA DE APRENDIZAJE

- **Denominación del programa de formación:** Análisis y desarrollo de software.
- **Código del programa de formación:** 228118
- **Nombre del proyecto:** Construcción de software integrador de tecnologías orientadas a servicios.
- **Fase del proyecto:** Ejecución.
- **Actividad de proyecto:** Codificar los módulos del software.
- **Competencias:**

Técnicas:

220501096 - Desarrollar la solución de software de acuerdo con el diseño y metodologías de desarrollo.

- **Resultados de aprendizaje a alcanzar:**

Técnicos:

220501096-01 - Planear actividades de construcción del software de acuerdo con el diseño establecido.

220501096-04 - Codificar el software de acuerdo con el diseño establecido.

- **Duración de la guía:** 498 horas

Técnico: 498 horas.

Recomendaciones generales

Actividad de aprendizaje GA7-220501096-AA1 configurar herramientas de versionamiento para control de código

Duración: 40 horas.
Materiales de formación: para el desarrollo de esta actividad es importante la lectura y análisis de los componentes formativos: “Aplicación del paradigma orientado a objetos” e “Integración continua”.
Evidencias:
A continuación, se describen las acciones y las correspondientes evidencias que conforma la actividad de aprendizaje:

Agenda del día

1. Saludo de Bienvenida.
2. Recomendaciones Generales
3. Evidencias a desarrollar

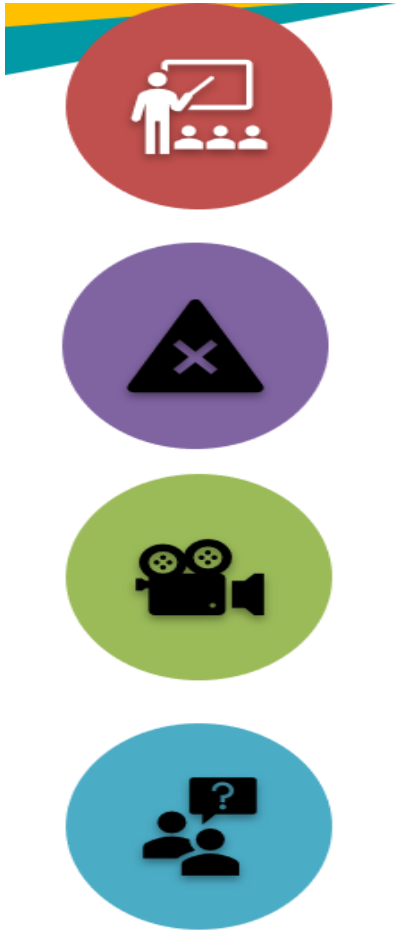
✓Evidencia de conocimiento: GA7-220501096-AA1-EV01 informe técnico de plan de trabajo para construcción de software.

✓Evidencia de desempeño: GA7-220501096-AA1-EV02 definir estándares de codificación de acuerdo a plataforma de desarrollo elegida.

✓Evidencia de conocimiento: GA7-220501096-AA1-EV03 identifica herramientas de versionamiento.

4. Resolución de Dudas e Inquietudes
5. Despedida Equipo de Instructores

Recomendaciones generales



1. Estudiar el material de la semana antes de la videoconferencia.
2. Evita distractores durante la sesión.
3. Toma nota de las dudas y preguntas que surjan durante la sesión y ten presente las que te hayan surgido durante el estudio del material.
4. Al final de la sesión, se asignarán 10 minutos para resolver todas las inquietudes que se hayan presentado.

Programación Orientada a Objetos (POO)

La **Programación Orientada a Objetos** (POO) es un paradigma de programación, es decir, un modelo o un estilo de programación que nos da unas guías sobre cómo trabajar con él. Se basa en el **concepto de clases y objetos.**



Lenguajes secuenciales como COBOL o procedimentales como Basic o C, se centraban más en la lógica que en los datos. Otros más modernos como Java, C# y Python, utilizan paradigmas para definir los programas, siendo la Programación Orientada a Objetos.



Claves de la Programación Orientada a Objetos

En este enfoque, cada una de estas entidades se convierte en un **objeto**, con **propiedades (datos)** y **comportamientos (funcionalidades)**. Por ejemplo, un objeto «Libro» puede tener atributos como el título, el autor y el año de publicación, así como métodos para obtener información sobre el libro, prestarlo o devolverlo a la biblioteca.

un objeto «Usuario» podría enviar un mensaje al objeto «Libro» para solicitar su préstamo, y este último respondería actualizando su estado interno.

```
10    * @author z
11    */
12   public class Producto {   // Clase
13
14       private int  codigo;   // atributos
15       private String nombre; // atributos
16       private double precio; // atributos
17
18       public void obtenerPrecio(){   // método
19           System.out.println("El precio es" + precio);
20       }
21
22       public void calcularTotal(){// método
23           double total = precio*5000;
24       }
25
26   }
27
```

Donde la línea 12 del código **representa la estructura de la clase** que es de acceso público y su nombre es Producto.

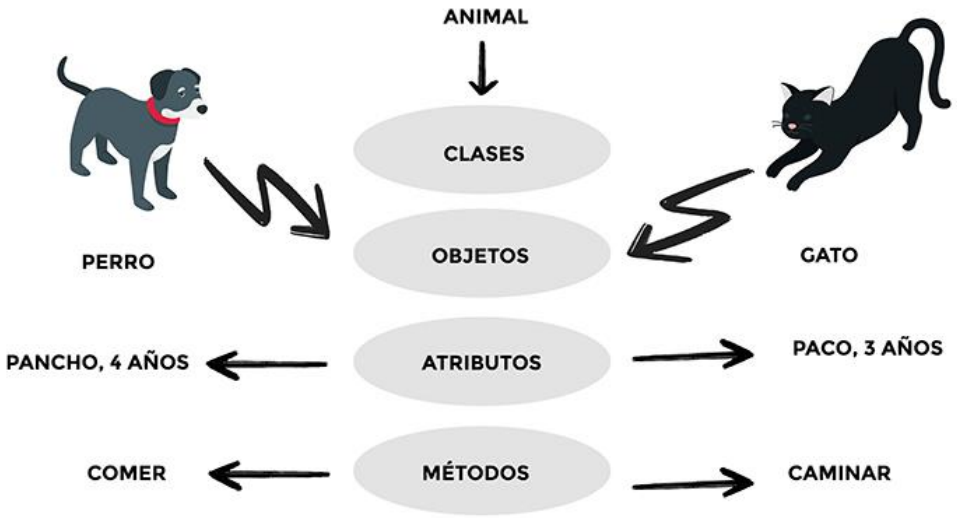
De la línea 14 a 16 **representa los atributos de la clase.**

A partir de la línea 18 **se encuentran los métodos**.

Clases, objetos e instancias

Una **clase** es una plantilla. Define de manera genérica cómo van a ser los objetos de un determinado tipo.

Un **ejemplo sencillo de un objeto,** como decíamos antes, podría ser un animal. Un animal tiene una edad, por lo que creamos un nuevo atributo de 'edad' y, además, puede envejecer, por lo que definimos un nuevo método.



Tipo	Formato	Descripción
Números Enteros		
byte	8 bit	Entero de un byte
short	16 bit	Entero corto
int	32 bit	Entero
long	64 bit	Entero Largo
Números Reales		
float	32 bit	Flotante simple
double	64 bit	Flotante doble
Cadenas de Texto		
char	16 bit	Un solo carácter
String	textos	Textos
Otros		
boolean	True o false	Valor booleano falso o verdadero

Principios de la Programación Orientada a Objetos

La encapsulación

Contiene **toda la información importante de un objeto dentro del mismo** y solo expone la información seleccionada al mundo exterior.

La abstracción

Es cuando **el usuario interactúa solo con los atributos y métodos seleccionados de un objeto**, utilizando herramientas simplificadas de alto nivel para acceder a un objeto complejo.

La herencia

Define **relaciones jerárquicas entre clases**, de forma que atributos y métodos comunes puedan ser reutilizados. Las clases principales extienden atributos y comportamientos a las clases secundarias.

El polimorfismo

Consiste en **diseñar objetos para compartir comportamientos**, lo que nos permite procesar objetos de diferentes maneras.



public:

Indica que los atributos son accesibles desde cualquier lugar del programa. No hay restricciones en el alcance de los miembros de datos públicos.



private:

Indica que los atributos sólo son accesibles dentro de la clase en la que se declaran.



protected:

Indica que los atributos son accesibles dentro del mismo paquete o sub-clases en paquetes diferentes. Al heredar sí se puede usar desde la clase derivada.

Implementar POO en diferentes lenguajes

Java:

Define clases utilizando la palabra clave «class» y especifica atributos y métodos dentro de ellas.
Utiliza la herencia con la palabra clave «extends» para crear subclases que hereden características de una clase base.
Aprovecha las interfaces para definir contratos que las clases deben implementar.
Aprovecha el polimorfismo para tratar objetos de diferentes clases de manera uniforme, utilizando métodos con la misma firma en diferentes clases.
Utiliza la encapsulación para ocultar los detalles internos de los objetos y proporcionar una interfaz pública para interactuar con ellos.

C#:

Define clases utilizando la palabra clave «class» y especifica atributos y métodos dentro de ellas.
Utiliza la herencia con los operadores de acceso «public» y «protected» para establecer la visibilidad de los miembros heredados.
Aprovecha las interfaces para definir contratos que las clases deben implementar.
Utiliza el polimorfismo utilizando la palabra clave «virtual» para los métodos en la clase base y «override» en las subclases.
Utiliza los modificadores de acceso «public», «private» y «protected» para lograr encapsulación y controlar el acceso a los miembros de una clase.

C++:

Define clases utilizando la palabra clave «class» y especifica atributos y métodos dentro de ellas.
Utiliza la herencia con los operadores de acceso «public», «private» y «protected» para establecer la visibilidad de los miembros heredados.
Aprovecha las funciones virtuales y la herencia múltiple para lograr polimorfismo.
Utiliza la encapsulación con los modificadores de acceso «public», «private» y «protected» para controlar el acceso a los miembros de una clase.

Python:

Define clases utilizando la palabra clave «class» y especifica atributos y métodos dentro de ellas.
Aprovecha la herencia utilizando la clase base entre paréntesis al definir una clase.
Utiliza la herencia múltiple y el polimorfismo de manera natural, ya que Python lo permite.
Aprovecha las propiedades y los decoradores para implementar la encapsulación y controlar el acceso a los atributos de una clase.

Entornos de desarrollo Java

ID	Logo	Enlace
NetBeans		https://netbeans.apache.org/download/index.html
Eclipse		http://www.eclipse.org/downloads/
JBlue		https://bluej.org/versions.html

SENA, 2021.

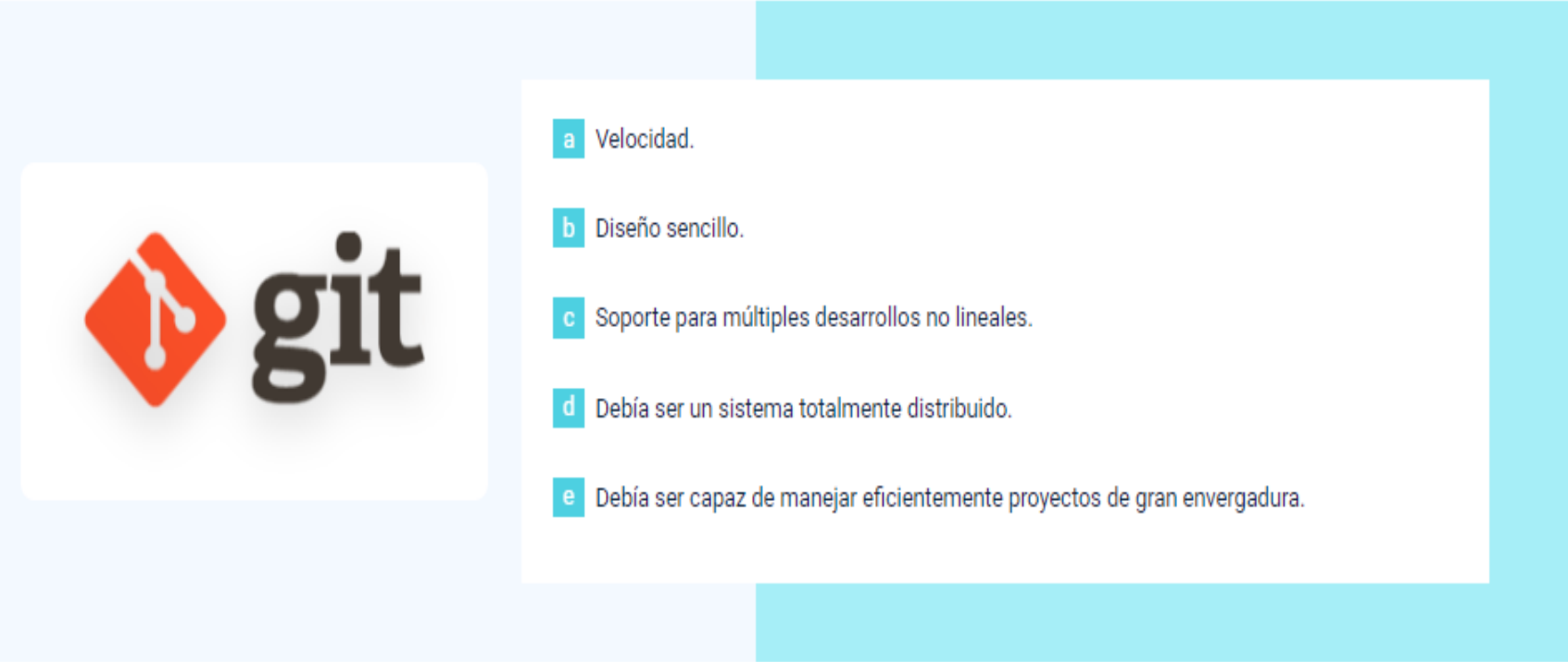
Sistemas de control de versiones

El proceso de hacer copias de seguridad y crear versiones de los archivos que se consideran importantes es una actividad muy común, así, por medio del uso de un sistema de control de versiones se obtienen gran cantidad de posibilidades entre las que se encuentran:



Sistema de control de versiones - Git

Git es un sistema de control de versiones que nació en el año 2005 producto de las necesidades que afrontaba la comunidad del kernel de Linux de encontrar un mecanismo que les permitiera soportar las actualizaciones y mantenimientos que requería este gran proyecto. Según Git (2021), algunos de los objetivos trazados que debía cumplir este nuevo mecanismo incluían:



diferencia de otros sistemas de control de versiones, posee un sistema de almacenamiento de información en los que cada versión posee una copia de instantáneas en miniatura, es decir, ofrece la posibilidad de trabajo aun cuando no se tiene conectividad al servidor central, ya que localmente se posee toda la información requerida incluso si se desea hacer procesos de recuperación a versiones anteriores.

Evidencia de conocimiento: GA7-220501096-AA1-EV01 informe técnico de plan de trabajo para construcción de software

Con base en las características del software a desarrollar, realice un informe técnico especificando herramientas y tecnologías de versionamiento a utilizar según lo visto en componentes Integración continua.

Elementos para tener en cuenta en el documento:

- Se deben seguir las normas básicas de presentación de un documento escrito, es decir el documento debe tener como mínimo una portada, introducción, objetivo, selección de herramientas de versionamiento a utilizar.



Lineamientos generales para la entrega de la evidencia:

- **Productos para entregar:** documento informe técnico
- **Extensión:** Libre.
- **Formato:** PDF.
- Para hacer el envío de la evidencia remítase al área de la actividad correspondiente y acceda al espacio:

informe técnico de plan de trabajo para construcción de software GA7-220501096-AA1-EV01.

Criterios de evaluación

- Interpreta el informe de diseño para definir el plan de trabajo en la construcción del software.
- Selecciona y configura herramientas de desarrollo de acuerdo con las condiciones del software a construir.
- Define estándares de codificación de acuerdo con las reglas de la plataforma de desarrollo seleccionada.

2. LISTA DE CHEQUEO

No.	VARIABLES/INDICADORES DE LOGRO	CUMPLE		Observaciones
		SI	NO	
1.	Diferencia con claridad versionamientos locales y remotos			30%
2.	Selecciona y describe herramientas de desarrollo a utilizar para la codificación del software.			30%
3.	Selecciona y describe herramientas de versionamiento			30%
4.	Presenta el informe cumpliendo con las normas básicas de presentación de documentos.			10%

Ejemplo

1. Portada
2. Introducción
3. Objetivo
4. Diferencia entre Versionamiento Local y Remoto
5. Selección de Herramientas de Desarrollo para la Codificación del Software
6. Selección y Descripción de Herramientas de Versionamiento
7. Conclusión

Evidencia de desempeño: GA7-220501096-AA1-EV02 definir estándares de codificación de acuerdo a plataforma de desarrollo elegida

Con base en las características del software a desarrollar realice un informe técnico con el estándar de codificación a utilizar para el desarrollo del software según lo visto en el componente "Aplicación del paradigma orientado a objetos".



Elementos para tener en cuenta en el documento:

- Se deben seguir las normas básicas de presentación de un documento escrito, es decir el documento debe tener como mínimo una portada, introducción, objetivo, estándares de codificación como nombramiento de variables, declaración de clases, declaración de métodos etc.

Lineamientos generales para la entrega de la evidencia:

- **Productos para entregar:** documento informe técnico
- **Extensión:** Libre.
- **Formato:** PDF.
- Para hacer el envío de la evidencia remítase al área de la actividad correspondiente y acceda al espacio: definir estándares de codificación de acuerdo a plataforma de desarrollo elegida GA7-220501096-AA1-EV02.

Criterios de evaluación

- Interpreta el informe de diseño para definir el plan de trabajo en la construcción del software.
- Selecciona y configura herramientas de desarrollo de acuerdo con las condiciones del software a construir.
- Define estándares de codificación de acuerdo con las reglas de la plataforma de desarrollo seleccionada.

2. LISTA DE CHEQUEO

No.	VARIABLES/INDICADORES DE LOGRO	CUMPLE		Observaciones
		SI	NO	
1.	Incluye en el informe Estándares de codificación según lenguaje seleccionado.			30%
2.	Específica nomenclatura para nombramiento de variables y constantes.			30%
3.	Específica nomenclatura para nombramientos de clases, métodos, comentarios			30%
4.	Presenta el informe cumpliendo con las normas básicas de presentación de documentos.			10%

Ejemplo

1. Portada
2. Introducción
3. Objetivo
4. **Estándares de Codificación**
    - 4.1. Nombramiento de Variables
    - 4.2. Declaración de Clases
    - 4,3. Declaración de Métodos
    - 4.4. Declaración de Constantes
    - 4.5. Comentarios
    - 4.6. Estándares Adicionales
7. Conclusión

Evidencia de conocimiento: GA7-220501096-AA1-EV03 identifica herramientas de versionamiento

Tomando como referencia el componente formativo “Integración continua”, realice una tabla con las diferencias entre el sistema de control de versionamiento git local y git remoto.

Elementos para tener en cuenta en el documento:

- Se deben seguir las normas básicas de presentación de un documento escrito, es decir el documento debe tener como mínimo una portada, introducción, objetivo, tablas con diferencias y características y comandos de git local y git remoto.
- Realice una tabla con la descripción de los comandos básicos de git remoto y git local



Lineamientos generales para la entrega de la evidencia:

- **Productos para entregar:** documento informe técnico
- **Extensión:** Libre.
- **Formato:** PDF.
- Para hacer el envío de la evidencia remítase al área de la actividad correspondiente y acceda al espacio:

identifica herramientas de versionamiento GA7-220501096-AA1-EV03.

Criterios de evaluación

• Selecciona y configura herramientas de versionamiento para el control de cambios en el código.

2. LISTA DE CHEQUEO

No.	VARIABLES/INDICADORES DE LOGRO	CUMPLE		Observaciones
		SI	NO	
1.	Selecciona herramientas de versionamiento según su tipo.			30%
2.	Describe las características de herramientas de versionamiento local.			30%
3.	Describe las características de herramientas de versionamiento remoto.			30%
4.	Presenta el informe cumpliendo con las normas básicas de presentación de documentos.			10%

Ejemplo

1. Portada
2. Introducción
3. Objetivo
4. **Diferencias entre Git Local y Git Remoto**

Aspecto	Git Local	Git Remoto

5. **Características de Git Local y Git Remoto**

Git Local	Git Remoto

6. **Comandos Básicos de Git Local**

Comando	Descripción

7. **Comandos Básicos de Git Remoto**

Comando	Descripción

8. Conclusión

Plazo Actividades:

16 al 21 de Septiembre



Dudas e inquietudes



1. Pedir la Palabra levantado la mano.
2. Iniciar indicando el nombre del instructor a quien se dirige la pregunta.
3. Ser muy exacto y breve en lo que se pregunta.

G R A C I A S

Línea de atención al ciudadano: 01 8000 910270
Línea de atención al empresario: 01 8000 910682

@SENAComunica

www.sena.edu.co