



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

Producción y compilación del contenido digital Inst. Angélica M. Triana
Solo fines académicos



www.sena.edu.co

Contenido

Acta de inicio

Estándares de codificación (código,
reglas de nombrado para variables,
métodos, atributos, clases, tablas,
campos, indentación)

Actividad de la sesión

Objetivos

Obtener los conocimientos propios para el manejo de herramientas informáticas.

Creecer en mi formación profesional integral (ser, saber, hacer).

Cumplir con los lineamientos de la formación.



Resultado de aprendizaje

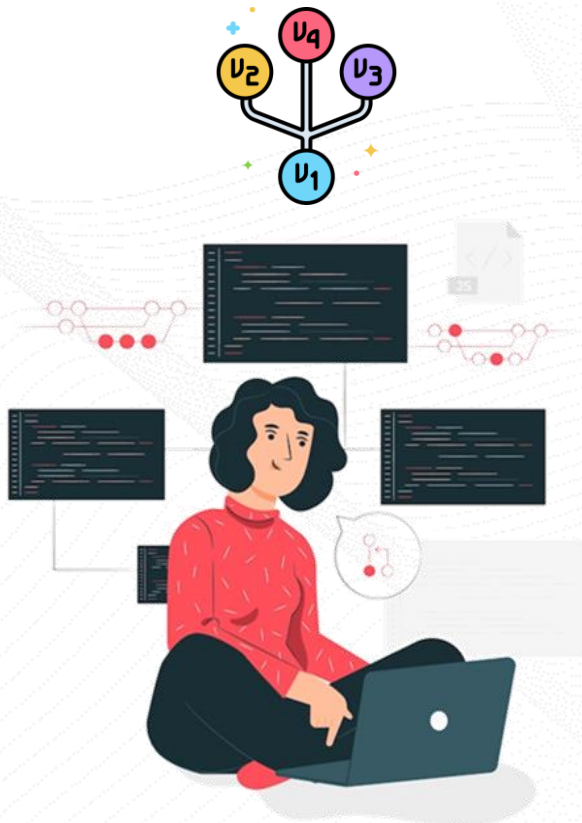


Competencia: Desarrollar la solución de software de acuerdo con el diseño y metodologías de desarrollo.



Resultado de aprendizaje:

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



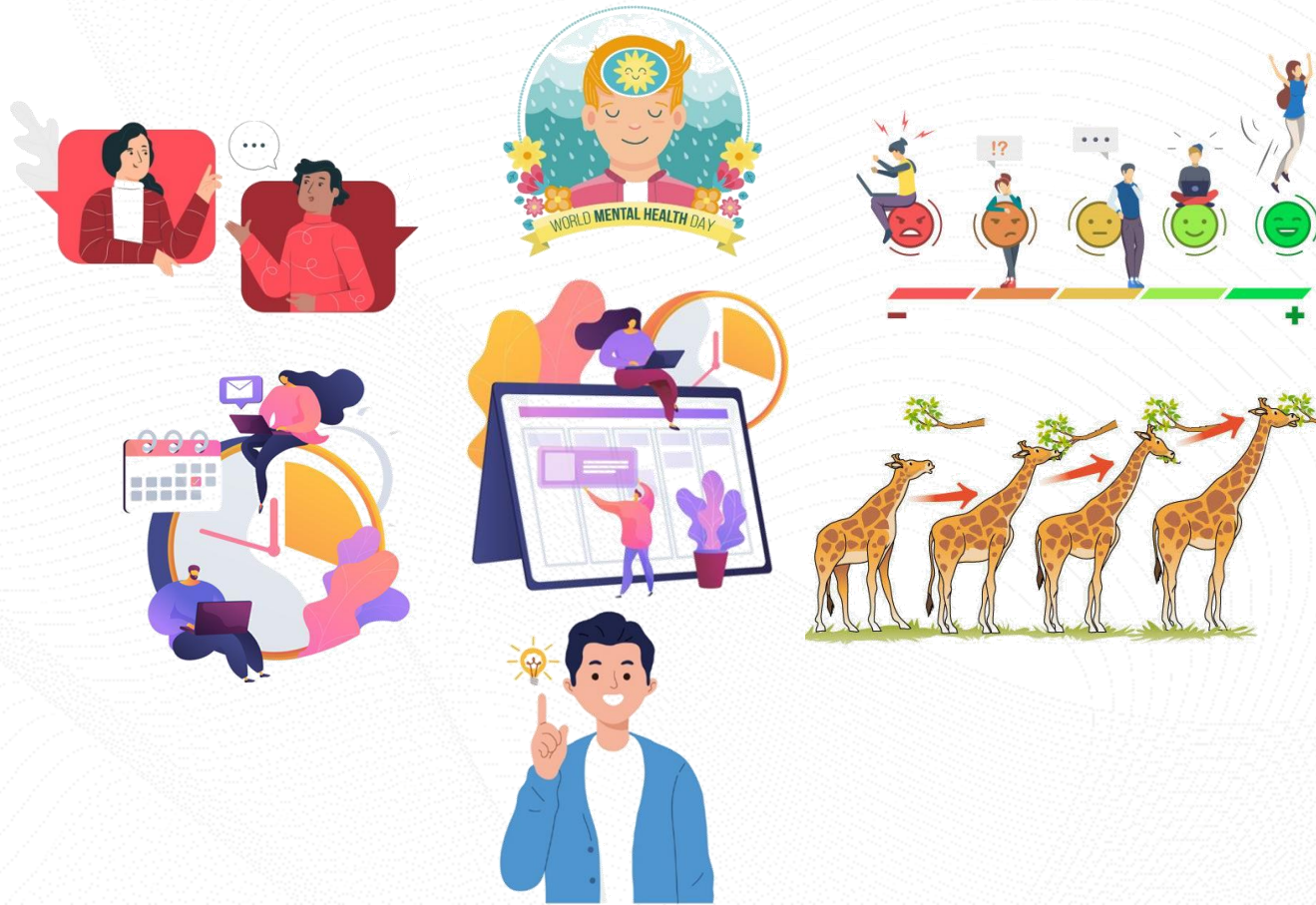
Primer momento



Acta de inicio




Documento donde se establecen los principales lineamientos académicos y comportamentales para el óptimo desarrollo de la formación profesional integral.



Puntos principales

- **Debido proceso:** Llamado de atención verbal -> llamado escrito + oportunidad de mejora -> Comité + plan de mejoramiento.
- Puntualidad – 3 llegadas tarde o inasistencia
- Evitar actitudes de somnolencia
- Enviar excusa justificada máximo 2 días hábiles directamente al instructor
- Adelantarse a las sesiones con sus compañeros
- Respeto – lenguaje, expresiones, presentación (*si aplica uniforme)
- No comer en el ambiente –no paquetes, chicles
- Recoger los residuos –mantener ambiente limpio
- Contestar el celular fuera del ambiente
- Autoría y entrega de las evidencias en los tiempos dados
- Todas las evidencias igual o mayor a una calificación del 70%
- No jugar en el ambiente de **formación**
- Escuchar música con audífonos solo si el instructor autoriza
- **Al ingresar revisar estado de los equipos y ambiente y antes de salir también!**

Reglas de uso de los equipos

The background of the slide features a man with dark, curly hair and a beard, wearing a dark purple or black short-sleeved shirt. He is standing in what appears to be a gym or a training facility, with metal railings and a wooden floor visible in the background. The lighting is somewhat dim, creating a focused and professional atmosphere.

La primera regla del ambiente de formación es:

No instalar programas o alterar la configuración del equipo sin autorización.

La segunda regla del ambiente de formación es:

No instalar programas o alterar

La tercera regla del ambiente de formación es:

No instalar programas o alterar la configuración del equipo sin autorización.

Herramientas



Medio para compartir los materiales de formación:

- OneDrive (correo @soyse^{na}.co)  OneDrive o link **Drive** gmail.
- Microsoft Team  o **meet** de gmail.

Entrega de las evidencias de producto:

- Zajuna (en espera), por ahora: drive o correo.

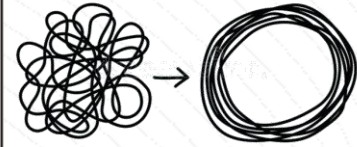
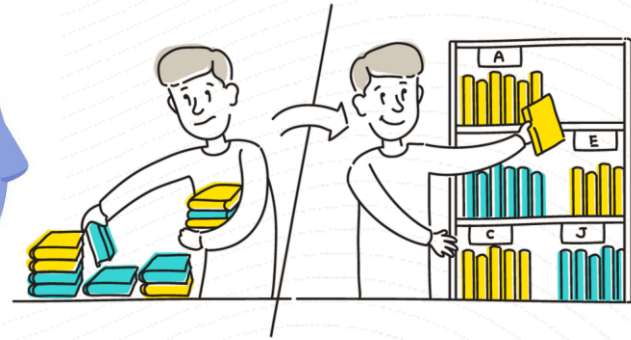
Software a utilizar:

- Suite ofimática, herramientas online - drive
- Visual Studio Code
- Navegador-Browser
- Cursos gratuitos online

Hardware:

- USB, tarjetas SD con adaptador, celular *Copias de seguridad.

Reflexión inicial



¿Buenas
prácticas?

Convención
o
Estandarización

Definición de estándar, convención o guía de código



*“Las convenciones de código son un **conjunto de directrices** para un lenguaje de programación específico que recomiendan estilo de programación, prácticas y métodos para cada aspecto de un programa escrito en ese lenguaje”.*

Recuperado de <https://dcodingames.com/convenciones-de-codigo/>

*“Las **guías de código** (también llamadas estándares de código o estilos de programación) es el nombre que se le da al conjunto de **normas usadas para escribir código fuente**, estas son regularmente dependientes del lenguaje de programación que se haya elegido”.*

Recuperado de https://codigofacilito.com/articulos/guia_codigo

*“Piense en los estándares de codificación como un **conjunto de reglas, técnicas y mejores prácticas para crear un código más limpio, más legible y más eficiente con errores mínimos**. Ofrecen un formato uniforme que los ingenieros de software pueden usar para crear código sofisticado y altamente funcional”. Recuperado de <https://www.browserstack.com/guide/coding-standards-best-practices#:~:text=Practices%20To%20Follow-.What%20are%20Coding%20Standards%3F,sophisticated%20and%20highly%20functional%20code.>*

Ventajas de implementar estándares de codificación



Programadores nombrando variables



- Ofrece uniformidad al código creado por diferentes ingenieros. (Comunicación entre el equipo de desarrollo)
- Permite la creación de código reutilizable.
- Facilita la detección de errores. (Calidad del programa)
- Haga que el código sea más simple, más legible y más fácil de mantener.
- Aumenta la eficiencia del programador y genera resultados más rápidos.

Cada programador tiene su propio estilo por ello he aquí criterios que ayudaran a mejorar a la hora de escribir código:

Estándares de codificación y mejores prácticas a seguir (parte 1)



Elija estándares de codificación específicos de la industria:

Las mejores prácticas y estándares de codificación **varían según** la industria para la que se crea un **producto específico**. Los estándares requeridos para el software de codificación para automóviles de lujo serán diferentes de los del software de codificación para juegos.

Centrarse en la legibilidad del código. El código legible es fácil de seguir, optimiza el espacio y el tiempo. Aquí hay algunas maneras de lograrlo:

- Escribe la menor cantidad de líneas posible.
- Use convenciones de **nomenclatura** apropiadas.
- Segmenta bloques de código en la misma sección en párrafos.
- Use **sangría** para marcar el comienzo y el final de las estructuras de control.
- No use funciones largas. Idealmente, una sola función debería llevar a cabo una sola tarea.
- Utilice el principio DRY (**Don't Repeat Yourself**). Automatice las tareas repetitivas cuando sea necesario. La misma pieza de código no debe repetirse en el script.

Estándares de codificación y mejores prácticas a seguir (parte 2)



- **Evite el anidamiento** profundo. Demasiados niveles de anidamiento hacen que el código sea más difícil de leer y seguir.
- Escriba con **mayúscula las palabras especiales de SQL** y los nombres de funciones para distinguirlos de los nombres de tablas y columnas.
- Evita las largas colas. Es más fácil para los humanos leer bloques de líneas que son horizontalmente cortos que verticalmente largos.

Estandarizar encabezados para diferentes módulos:

Es más fácil comprender y **mantener el código** cuando los encabezados de diferentes módulos se alinean con un formato singular. Por ejemplo, cada encabezado debe contener:

Nombre del módulo, Fecha de creación, Nombre del creador del módulo, Historia de la modificación, Resumen de lo que hace el módulo, Funciones en ese módulo, Variables a las que accede el módulo.

Estándares de codificación y mejores prácticas a seguir (parte 3)



No utilice un único identificador para múltiples propósitos:

Asigne **un nombre a cada variable que describa claramente su propósito**. Naturalmente, a una sola variable no se le pueden asignar múltiples valores o usarse para numerosas funciones. Esto confundiría a todos los que lean el código y dificultaría la implementación de futuras mejoras.

Asigne siempre nombres de variables únicos.

Convierta las copias de seguridad diarias en un instinto:

Múltiples eventos pueden desencadenar la pérdida de datos: bloqueo del sistema, batería agotada, falla del software, daño del hardware, etc. Para evitar esto, guarde el código diariamente y después de cada modificación, sin importar cuán minúscula sea. Realice una copia de seguridad del flujo de trabajo en TFS (Team Foundation Version Control – Microsoft), SVN (Subversion – open source) o cualquier otro mecanismo de **control de versiones** (GitHub).

Estándares de codificación y mejores prácticas a seguir (parte 4)



Intente formalizar el manejo de excepciones

'Excepción' se refiere a problemas, problemas o eventos poco comunes que ocurren cuando se ejecuta el código e **interrumpen el flujo normal de ejecución**. Esto detiene o finaliza la ejecución del programa, que es un escenario que debe evitarse. Sin embargo, cuando ocurran, use las siguientes técnicas para minimizar el daño a la ejecución general en términos de tiempo y esfuerzo del desarrollador:

- Mantenga el código en un bloque **try-catch**.
- Asegúrese de que la **recuperación** automática se haya activado y se pueda utilizar.
- Considere que podría ser un problema de lentitud del software o de la red. Espere unos segundos a que aparezcan los elementos necesarios.
- Utilice el análisis de registros en tiempo real.

Estándares de codificación y mejores prácticas a seguir (parte 5)



Dejar comentarios y priorizar documentación:

No asuma que solo porque todos los demás que ven el código son desarrolladores, instintivamente lo entenderán sin aclaración. Los desarrolladores son humanos, y es mucho más fácil para ellos **leer comentarios que describen la función del código** en lugar de escanear el código y hacer especulaciones. Por supuesto, esto solo es necesario cuando el propósito del código no es evidente. **No se moleste en dejar comentarios en el código que se explica por sí mismo.**

Al elegir estándares, piense en Cerrados vs. Abiertos

Los cerrados son específicos para una organización o empresa. Los estándares abiertos cambian rápidamente, lo que puede hacer que sea difícil mantenerse al día. Sin embargo, los estándares cerrados son mejores para las industrias críticas para la seguridad porque imponen uniformidad entre equipos, organizaciones y proveedores. En otras palabras, ofrecen una referencia confiable que exige el cumplimiento de un conjunto de requisitos obligatorios. Tenga en cuenta que todo el **código debe probarse exhaustivamente en navegadores y dispositivos reales.**

Normas y directrices de codificación, aspectos comunes de los estándares



Las normas y directrices de codificación garantizan que el software sea:

- Seguro: se puede utilizar sin causar daño.
- Confiable: Funciona como debe, siempre.
- Comprobable: Se puede probar a nivel de código.
- Mantenable: se puede mantener, incluso a medida que crece su base de código.
- Portátil: Funciona igual en todos los entornos.

Aspectos comunes de los estándares de codificación:

- Convenciones de nomenclatura
- Denominación y organización de archivos
- Formato y sangría
- Comentarios y documentación
- Clases, funciones e interfaces
- Uso de punteros y referencias

¿Hace parte de los requerimientos no funcionales?

Convenciones comunes de la codificación (parte 1)

Nombres de variables y procedimientos:

Los nombres de variables y procedimientos definidos deben **tener significado**.

Estos nombres deben ser auto explicativos y representativos con respecto a su propósito. La información nombrada sin sentido solo logra dificultar la lectura y comprensión del código.

Correcto:

```
string direccion;  
int usuario;
```

Incorrecto:

```
string nom;  
string dir;  
int pass;
```

Indentación:

Básicamente la indentación es usada para tener una mejor visibilidad en el diseño de un programa, nos muestra las líneas que son subordinadas de otras líneas. La mayoría de los lenguajes utilizan llaves para indentar o **delimitar bloques de código**.

```
public class Ejemplo  
{  
    0 referencias  
    public Ejemplo()  
    {  
        //  
        // TODO: Agregar aquí la lógica del constructor  
        //  
        "Alta Usuario"  
        "Asignación Roles"  
        "Privilegios"  
    }  
}
```

Convenciones comunes de la codificación (parte 2)

Espaciado:

La mayoría de los lenguajes ignoran los espacios en blanco, sin embargo, **mejora la legibilidad y comprensión**, agrega espacios en los siguientes lugares: Entre operadores. Antes y después del operador de asignación.

Funciones coherentes:

Cada función debe ser diseñada para una tarea simple.

```
def mi_funcion(nombre, apellido):  
    nombre_completo = nombre, apellido  
    print nombre_completo
```

Los **parámetros** de las funciones deben **estar separados por un espacio después de cada coma.**

Comentarios del Código:

Las secciones de código deben tener comentarios donde se defina su función de forma clara de tal forma que pueda ser **documentada y entendida posteriormente.**

Correcto:

```
//Este es un comentario de ejemplo que está alineado  
string MensajeCompleto = "Hola" + nombre;  
MessageBox.Show(MensajeCompleto);
```

Incorrecto:

```
//Este es un comentario de ejemplo que NO está alineado  
string MensajeCompleto = "Hola" + nombre;  
MessageBox.Show(MensajeCompleto);
```

Convenciones comunes de la codificación (parte 3)

Llaves:

Utiliza siempre las llaves, incluso para los bloques que contienen sólo una sentencia. Esto elimina una fuente común de bugs y facilita el mantenimiento

Longitudes y saltos de línea:

Escribe una sentencia por línea solamente, **trata de evitar que la línea no sobrepase los 80** caracteres como máximo, si es necesario aplica un salto de línea de manera notoria: Luego del operador de asignación. En expresiones aritméticas y lógicas antes de un operador.

```
using System;

namespace inline_function
{
    class Program
    {
        static void Main(string[] args)
        {
            void sum(int a, int b)
            {
                Console.WriteLine(a + b);
            }
            sum(7, 11);
        }
    }
}
```

En las invocaciones de método luego del paréntesis de apertura.

Paréntesis:

Utiliza los paréntesis en expresiones que impliquen distintos operadores para evitar problemas con la precedencia, pues, aunque creas que la precedencia es clara, puede que esto no sea así para otros.

Tipos de notación

Consulte el enlace de la lectura: <https://www.neoguias.com/tipos-notacion-nombres/>

De acuerdo con lo anterior, señale con una equis (x) según corresponda:

nombre	Camel Case	Pascal Case	Snake Case	Kebab Case
valorDescuento				
valor-descuento				
ValorDescuento				
valcular_descuento				

En los lenguajes Orientados a Objetos se espera que las clases sean en mayúsculas y que los métodos y atributos (variables) en minúsculas, en algunos casos si es una constante el nombre de esta sería en Mayúscula. Este tipo de notación también se refleja en los diagramas UML (ej.: diag. De clases, de secuencia).

¿Donde encontrarlas?

Cada guía de código depende del lenguaje de programación a utilizar y en internet existen distintas fuentes en donde las puedes encontrar, pero se recomienda las guías que son creadas por la comunidad, ya que estas crecen y se mejoran día tras día gracias a los aportes de las personas que contribuyen con el proyecto.

A continuación algunas guías de estilo:

- Python <https://peps.python.org/pep-0008/> En español: <https://recursospython.com/pep8es.pdf>
- PHP <https://www.mclibre.org/consultar/php/otros/guia-estilo.html> y <https://latteandcode.medium.com/estandares-psr-para-escribir-codigo-php-acc68d97f6d3>
- Java <https://silo.tips/download/estandares-de-codificacion-java> Cerrada: <https://www.juntadeandalucia.es/servicios/madeja/contenido/libro-pautas/15>
- C# <https://docs.microsoft.com/es-es/dotnet/csharp/fundamentals/coding-style/coding-conventions>
- Kotlin <https://developer.android.com/kotlin/style-guide?hl=es-419>

Ejemplo



Estandarizar el código de “hola mundo” dependiendo del lenguaje:

Python	PHP	Java	JavaScript	C#
<pre>print ("Hola Mundo!")</pre>	<pre><?php echo "Hola Mundo!"; ?></pre>	<pre>public class Programa { public static void main(String[] args) { System.out.println("Hola mundo!"); } }</pre>	<pre>console.log("Hello world!"); //petición-respuesta</pre>	<pre>using System; class Programa{ public static void Main(string[] args) { Console.WriteLine("Hola Mundo!"); } }</pre>


¿El código
podría
ejecutarse de
esta forma?

Ejemplo: solución




Estandarizar el código de “hola mundo” dependiendo del lenguaje:

Python	PHP	Java	JavaScript	C#
<code>print ("Hola Mundo!")</code>	<code><?php echo "Hola Mundo!"; ?></code>	<code>public class Programa { public static void main(String[] args) { System.out.println("Hola mundo!"); } }</code>	<code>console.log("Hello world!"); //petición-respuesta console.log("Hello world!");</code>	<code>using System; class Programa{ public static void Main(string[] args) { Console.WriteLine("Hola Mundo!"); } }</code>



```
/*  
 * To change this license header, choose License Headers in Project  
Properties.  
 * To change this template file, choose Tools | Templates  
 * and open the template in the editor.  
 */  
package javaapplication3;  
  
/**  
 *  
 * @author Family  
 */  
public class Programa {  
  
    /**  
     * @param args the command line arguments  
     */  
    public static void main(String[] args) {  
        // TODO code application logic here  
        System.out.println("Hola mundo!");  
    }  
}
```



```
using System;  
  
namespace HolaMundo  
{  
    class Program  
    {  
        static void Main(string[] args)  
        {  
            Console.WriteLine("Hola Mundo!");  
        }  
    }  
}
```

¿Por qué tener estándares de codificación?

- Reducir el costo de mantenimiento del software es la razón citada con más frecuencia para seguir los estándares de codificación.
- El 80% del costo de por vida de una pieza de software se destina al mantenimiento.
- Los estándares de codificación aumentan sustancialmente la legibilidad.
- Cualquier miembro de un desarrollo debe poder leer el código de otro miembro.



Actividad de la sesión



Actividad de la sesión



En grupos de proyecto,

Investigar y desarrollar los siguientes puntos que serán socializados en la sesión:

- 1) ¿Qué es una palabra reservada en programación? Pueden decir como se identifica independientemente del lenguaje
- 2) ¿Qué es concatenar?
- 3) ¿Qué son lenguajes tipados y no tipados?, ¿qué significa que sea fuertemente tipado?
- 4) ¿Qué es un script?
- 5) Consulten las guías de estilo para los diferentes lenguaje de programación (revisar slides anteriores) y diligencie la siguiente tabla para cada lenguaje:



Actividad de la sesión

Tabla a diligenciar para cada lenguaje de programación: Java, JavaScript, PHP, Python, C#

Estándar de codificación XYZ	Descripción	Ejemplo
Nombres de clases y métodos		
Sangría y legibilidad del código		
Declaración de variables y constantes		
Comentarios		

En cada estándar mostrar un ejemplo al respecto, puede ser una captura de pantalla.

Se expondrá los resultados de cada grupo en la sesión.

**WRITES LONG COMPLICATED
SOFTWARE CODE**



NO COMMENTS



G R A C I A S

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



www.sena.edu.co

Referencias



Cantoral, C. (2017). ¿Que es una guía de código? Recuperado de:
https://codigofacilito.com/articulos/guia_codigo

Bose, S. (2021). Coding Standards and Best Practices To Follow. Recuperado de
<https://www.browserstack.com/guide/coding-standards-best-practices#:~:text=Practices%20To%20Follow-,What%20are%20Coding%20Standards%3F,sophisticated%20and%20highly%20functional%20code.>

Kumar, S. (2019). Software coding and testing. Recuperado de
<https://www.slideshare.net/SandeepKumarNayak1/software-coding-and-testing>