PROCESO DE GESTIÓN DE FORMACIÓN PROFESIONAL INTEGRAL

FORMATO GUÍA DE APRENDIZAJE

1. IDENTIFICACIÓN DE LA GUIA DE APRENDIZAJE

- Denominación del Programa de Formación: Programación de Software
- Código del Programa de Formación: 233104
- Nombre del Proyecto Formativo: Soluciones de software para la gestión de organizaciones.
- Fase del Proyecto HACER:
- Actividad de Proyecto Formativo: Codificar la solución del software de acuerdo con el diseño generado.
- Competencia: Desarrollo de la solución de software.
- Resultados de Aprendizaje: Resolver procesos lógicos a través de la implementación de algoritmos y el lenguaje de programación seleccionado.
- Duración de la Guía de Aprendizaje (horas):
- HORAS TRABAJO DIRECTO: 110
- TRABAJO INDEPENDIENTE: 27

2. PRESENTACIÓN

- Este logro esperado busca que el aprendiz desarrolle habilidades para identificar y solucionar problemas mediante el pensamiento lógico. Implica comprender la lógica de los procesos, descomponerlos en pasos ordenados y traducirlos en algoritmos claros. Además, requiere que el aprendiz elija un lenguaje de programación adecuado según el contexto del problema. A través de la implementación de estos algoritmos, el aprendiz debe demostrar capacidad para codificar soluciones eficientes y funcionales. El objetivo es fomentar el razonamiento estructurado, la creatividad en la solución de problemas y la precisión técnica. Esto

es esencial en entornos laborales donde se requiere automatizar tareas, desarrollar software o analizar datos. También se valora la capacidad de probar, depurar y mejorar el código. En conjunto, este logro fortalece la empleabilidad del aprendiz en áreas tecnológicas y de desarrollo informático.

3. FORMULACIÓN DE LAS ACTIVIDADES DE APRENDIZAJE

- **Descripción de la(s) Actividad(es)**

3.1 Actividades de reflexión inicial:

Descripción de la actividad:

1) Situación problema real:

Muchos aprendices olvidan entregar tareas a tiempo, estudiar para exámenes o asistir a reuniones importantes debido a la falta de una herramienta de organización adaptada a sus necesidades escolares. Esto afecta su rendimiento académico y genera estrés.

2) Pregunta problema:

¿Cómo podemos diseñar un programa funcional que nos ayude a organizar nuestras actividades escolares mediante el uso de algoritmos y estructuras básicas de programación?

3) Actividad de aprendizaje para reflexionar sobre la situación problema:

Nombre: Foro inicial de reflexión: "¿Por qué olvidamos las tareas?"
Actividad:

- En equipos, los aprendices comparten experiencias donde olvidar una actividad escolar les causó problemas.
- Realizan una lluvia de ideas de posibles soluciones tecnológicas.
- Cada equipo redacta un breve diagnóstico de la situación, señalando causas y consecuencias del problema.

Agrupación: Equipos colaborativos

Ambiente requerido: PRESENCIAL: Área mínima: 70 mt2. 30 Computadoras, memoria RAM 8Gb o superior, procesador mayor o igual a 2.5 Ghz, Disco Duro de 512 GB o SSD de 250 GB o superior

Estrategias o técnicas didácticas activas: Foro inicial de reflexión: "¿Por qué olvidamos las tareas?"

Materiales de formación: Aplicación puzle,flipiity

Material de apoyo: Anexo 1 documento

Duración de la actividad: horas.

3.2 Actividades de contextualización e identificación de conocimientos necesarios para el aprendizaje:

Descripción de la actividad:

4) Contexto de la situación problema:

¿Quién tiene el problema y cuándo/dónde surge?
Aprendiz de formación técnica en programación. El problema ocurre durante su proceso de formación, cuando deben cumplir múltiples tareas, prácticas y evaluaciones.

¿Cuál es la causa del problema?
La falta de una herramienta de organización adaptada a su contexto educativo y hábitos personales. Proviene de una observación del comportamiento cotidiano de los aprendices, no de una investigación formal.

5) Actividad para la contextualización e identificación de conocimientos previos:

Nombre: "¿Qué sabemos sobre programación y lógica?"
Actividad:

- Cuestionario diagnóstico con preguntas sobre lógica proposicional, estructuras secuenciales, condicionales, cíclicas y arreglos.
- Socialización de respuestas en pequeños grupos.
- Mapa conceptual colectivo sobre lo que ya conocen y lo que deben reforzar.

Agrupación: Individual (cuestionario) y colaborativo (mapa conceptual)

Ambiente requerido:

Estrategias o técnicas didácticas activas: Trabajo colaborativo

Materiales de formación cmap tolos, miro, luicidchart

Material de apoyo: cuestionario

Duración de la actividad: horas.

3.3 Actividades de apropiación:

Descripción de la actividad:

6) Objetivos de aprendizaje:

Cognitivos:

- Comprender los fundamentos de la lógica proposicional y su aplicación en algoritmos.
- Conocer y aplicar estructuras de programación básicas: secuenciales, condicionales, cíclicas y arreglos.

Procedimentales:

- Diseñar algoritmos que resuelvan problemas cotidianos usando estructuras lógicas.
- Implementar programas funcionales en un lenguaje de programación básico (PSeInt, Python, etc.).

Actitudinales:

- Demostrar responsabilidad en el trabajo colaborativo.
- Mostrar iniciativa para resolver problemas reales a través de la programación.

7) Instrucciones paso a paso (modo imperativo, tercera persona, con objeto de aprendizaje y condición):

1. **Resuelva problemas de lógica proposicional**, aplicando conectores lógicos (y, o, no, si-entonces), cuando se enfrente a decisiones condicionales en la programación de tareas escolares.
2. **Utilice estructuras secuenciales**, al diseñar el flujo principal del programa, siempre que el proceso requiera una ejecución paso a paso sin condiciones.
3. **Implemente estructuras de control condicional**, cuando deba evaluar si una tarea debe ser mostrada, editada o eliminada, usando estructuras como `if`, `if-else`.
4. **Aplique estructuras cíclicas**, para recorrer listas de tareas o repetir menús de acciones, cuando el usuario decida seguir operando dentro del programa.
5. **Construya algoritmos con arreglos**, para almacenar múltiples tareas con atributos (nombre, fecha, estado), cada vez que necesite organizar datos similares y consultarlos por posición o índice.

8) Tipo de agrupación:

Parte del proyecto	Tipo de agrupación
Diagnóstico del problema	Colaborativo
Investigación de soluciones	Colaborativo
Diseño del algoritmo	Colaborativo
Codificación del programa	Individual
Pruebas y validación del programa	Individual
Presentación y retroalimentación final	Colaborativo

9) Productos que deberán elaborar:

- Documento de análisis del problema (equipo)
- Diagrama de flujo o pseudocódigo de la solución (equipo)
- Código fuente del programa funcional (individual)
- Prueba de escritorio documentada (individual)
- Presentación final del proyecto (equipo)

10) Temporización (proyecto de 4 semanas, 2 sesiones semanales de 2 horas):

Semana	Sesión	Actividades principales
1	1	Introducción al proyecto, reflexión sobre el problema, formación de equipos
	2	Diagnóstico de conocimientos previos, socialización, lluvia de ideas
2	3	Diseño del algoritmo: identificación de procesos lógicos y estructuras a usar
	4	Elaboración de diagramas de flujo y pseudocódigo
3	5	Codificación individual del programa en el lenguaje seleccionado
	6	Prueba de escritorio y validación de errores
4	7	Ajustes, mejora del código, preparación de presentación

Semana Sesión	Actividades principales
8	Presentación de productos finales y retroalimentación en equipos

¿Te gustaría que prepare una plantilla de evaluación o rúbrica para este proyecto?

Ambiente requerido:

Estrategias o técnicas didácticas activas:

Materiales de formación: Aplicación

Material de apoyo: anexo 1 documento

Evidencias de aprendizaje:

Instrumentos de evaluación:

Duración de la actividad: horas.

3.4 Actividades de Transferencia el Conocimiento:

11) Producto y desempeño esperado según conocimientos de proceso:

Producto final concreto:
Un programa funcional en PSeInt, Python o similar que permita registrar, visualizar, editar y eliminar tareas escolares, utilizando lógica proposicional, estructuras secuenciales, condicionales, cíclicas y arreglos. También debe incluir pseudocódigo o diagrama de flujo y una prueba de escritorio documentada.

Desempeño esperado del aprendiz:
El aprendiz debe ser capaz de:

- Identificar correctamente la lógica proposicional implicada en las condiciones del programa (por ejemplo, cuándo mostrar o no una tarea).
- Usar estructuras secuenciales para establecer el flujo base del programa (menús, opciones).
- Implementar estructuras condicionales para tomar decisiones dentro del algoritmo (mostrar tarea vencida, validar fechas, etc.).
- Aplicar estructuras cíclicas para repetir acciones o recorrer listas de tareas.
- Utilizar arreglos para almacenar y gestionar múltiples tareas con diferentes atributos.
- Documentar correctamente el algoritmo mediante pseudocódigo y/o diagramas de flujo.
- Validar su algoritmo mediante una prueba de escritorio que contemple casos comunes y extremos.

12) Instrucciones secuenciales (modo imperativo, formación por procesos):

1. **Analice la situación problema**, identificando las necesidades de organización académica que enfrenta el aprendiz, cuando no cuenta con herramientas de planificación digital.
2. **Defina los requerimientos del programa**, determinando qué funciones debe incluir (agregar tarea, consultar, eliminar, etc.), al planear la solución inicial de la agenda escolar.
3. **Formule condiciones lógicas simples y compuestas**, al establecer las reglas que debe cumplir el sistema para ejecutar determinadas acciones (por ejemplo, si la tarea está vencida, mostrar advertencia).
4. **Diseñe el algoritmo en pseudocódigo o diagrama de flujo**, representando el flujo de la solución con estructuras secuenciales, condicionales y cíclicas, cuando haya definido claramente el comportamiento esperado del programa.
5. **Implemente el algoritmo en el lenguaje de programación seleccionado**, asegurándose de declarar correctamente arreglos para almacenar tareas y usar estructuras de control adecuadas en cada parte del proceso.
6. **Ejecute una prueba de escritorio del programa**, aplicando diferentes casos de uso (tareas vencidas, tareas múltiples, errores de entrada), cuando el código esté estructurado y sin errores de sintaxis.
7. **Realice ajustes y mejoras al programa**, corrigiendo errores lógicos y optimizando el uso de estructuras, cuando los resultados de la prueba de escritorio no coincidan con los esperados.
8. **Presente el producto final del proyecto**, incluyendo el código funcional, pseudocódigo o diagrama de flujo, y la documentación de la prueba de escritorio, cuando haya validado su funcionamiento completo.

Descripción de la actividad:

Ambiente requerido:

Estrategias o técnicas didácticas activas:

Materiales de formación

Material de apoyo:

Evidencias de aprendizaje:

Instrumentos de evaluación:

Duración de la actividad: horas.

4. PLANTEAMIENTO DE EVIDENCIAS DE APRENDIZAJE PARA LA EVALUACIÓN EN EL PROCESO FORMATIVO.

Fase del proyecto formativo	Actividad del proyecto formativo	Actividad de Aprendizaje	Evidencias de Aprendizaje	Criterios de Evaluación	Técnicas e Instrumentos de Evaluación

5. GLOSARIO DE TÉRMINOS

Algoritmo
Secuencia ordenada de pasos para resolver un problema.

Arreglo
Estructura que almacena varios datos del mismo tipo en una sola variable.

Ciclo
Repetición de un bloque de código mientras se cumpla una condición.

Condicional
Instrucción que ejecuta diferentes acciones según se cumpla una condición.

Desarrollo de software
Proceso de creación de programas para resolver necesidades específicas.

Diseño de software
Planificación estructurada de cómo funcionará un sistema antes de programarlo.

Estructura secuencial
Ejecución de instrucciones una tras otra en orden.

Lógica proposicional
Uso de enunciados lógicos (verdadero/falso) en decisiones de un algoritmo.

Metodologías de desarrollo
Modelos que guían el proceso de creación de software (como cascada o ágil).

Pseudocódigo
Forma simple de escribir algoritmos sin usar sintaxis de un lenguaje específico.

Prueba de escritorio
Simulación manual del algoritmo para verificar su funcionamiento.

Programa
Conjunto de instrucciones que ejecuta una tarea mediante un lenguaje de programación.

Requerimientos funcionales
Funciones que debe cumplir un software desde el punto de vista del usuario.

Solución de software
Aplicación informática diseñada para resolver un problema concreto.

Validación
Comprobación de que un software funciona correctamente según lo esperado.

Referencias en norma APA (7.ª edición)

- Arends, R. I. (2014). *Aprendizaje y enseñanza*. McGraw-Hill Education.
- Díaz-Barriga, F., & Hernández, G. (2010). *Estrategias docentes para un aprendizaje significativo: Una interpretación constructivista*. McGraw-Hill.
- Johnson, D. W., & Johnson, R. T. (2013). *Aprendizaje cooperativo en el aula*. Paidós.
- Ministerio de Educación Nacional de Colombia (MEN). (2014). *Orientaciones para la implementación del enfoque por competencias en la formación profesional integral*. SENA. https://www.sena.edu.co
- Pozo, J. I. (2006). *Adquisición de conocimiento y aprendizaje escolar*. Morata.
- Polya, G. (2004). *Cómo plantear y resolver problemas*. Trillas.
- SENA – Servicio Nacional de Aprendizaje. (2013). *Guía metodológica para el diseño curricular por competencias laborales*. Bogotá: SENA. https://repositorio.sena.edu.co
- Slavin, R. E. (2020). *Psicología educativa: Teoría e investigación para la enseñanza*. Pearson Educación.
- Svinicki, M., & McKeachie, W. J. (2014). *Teaching Tips: Strategies, Research, and Theory for College and University Teachers*. Cengage Learning.
- Wirth, N. (2003). *Algoritmos + estructuras de datos = programas*. Prentice Hall.

7. CONTROL DEL DOCUMENTO

	Nombre	Cargo	Dependencia	Fecha
Autor (es)				

8. CONTROL DE CAMBIOS (diligenciar únicamente si realiza ajustes a la guía)

	Nombre	Cargo	Dependencia	Fecha	Razón del Cambio
Autor (es)					