PROCESO DE GESTIÓN DE FORMACIÓN PROFESIONAL INTEGRAL

FORMATO GUÍA DE APRENDIZAJE

1. IDENTIFICACIÓN DE LA GUIA DE APRENDIZAJE

- **Denominación del Programa de Formación:** PROGRAMACION DE SOFTWARE.
- **Código del Programa de Formación:** 233104 VERSION 2
- **Nombre del Proyecto Formativo**: 2714790 SOLUCIONES DE SOFTWARE PARA LA GESTIÓN DE ORGANIZACIONES.
- **Fase del Proyecto**: ANALISIS.
- **Actividad de Proyecto Formativo**: Codificar la solución del software de acuerdo con el diseño generado.
- **Competencia:** Desarrollar la solución de software de acuerdo con el diseño y metodologías de desarrollo.
- **Resultados de Aprendizaje:** RAP 1. Resolver procesos lógicos a través de la implementación de algoritmos y el lenguaje de programación seleccionado.
- **Duración de la Guía de Aprendizaje (horas):** 137 horas
- **HORAS TRABAJO DIRECTO:** 102
- **HORAS TRABAJO INDEPENDIENTE:** 35

2. PRESENTACIÓN

El aprendiz será capaz de analizar situaciones problemáticas y transformarlas en soluciones estructuradas mediante el uso del pensamiento lógico. Diseñará algoritmos que representen de forma ordenada y eficiente los pasos necesarios para resolver un problema. Implementará dichos algoritmos utilizando el lenguaje de programación seleccionado, aplicando correctamente su sintaxis y estructuras básicas. Además, validará el funcionamiento de las soluciones desarrolladas, realizando ajustes cuando sea necesario. Este logro le permitirá automatizar procesos, optimizar tareas y fortalecer su capacidad para desarrollar soluciones informáticas acordes a contextos laborales reales.

3. FORMULACIÓN DE LAS ACTIVIDADES DE APRENDIZAJE

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

3.1 Actividades de reflexión inicial:

Descripción de la actividad:

3.1.1 Situación problema (desde el enfoque por competencias laborales)

Contexto:

Una microempresa comercial (tienda de barrio, ferretería o papelería) presenta constantes inconsistencias en el control de su inventario: productos que aparecen agotados en el sistema pero están disponibles físicamente, registros duplicados, salidas sin validar cantidades y falta de alertas cuando el stock es bajo.

Esta situación afecta la atención al cliente y la toma de decisiones del administrador.

La empresa requiere una solución de software que permita controlar el inventario de manera confiable, aplicando lógica proposicional y algoritmos, utilizando estructuras secuenciales, condicionales, cíclicas y arreglos, de acuerdo con el diseño definido y el lenguaje de programación seleccionado.

3.1.2 Pregunta problema

¿Cómo diseñar e implementar una solución de software que permita controlar el inventario de una microempresa, validando entradas y salidas de productos y generando alertas oportunas, mediante la aplicación de lógica proposicional, algoritmos y estructuras de programación?

3.1.3 Actividad de reflexión inicial

“¿Por qué no cuadran los productos?”

Técnica didáctica activa: Análisis de caso con lluvia de ideas colaborativa

Actividad:

El instructor presenta un caso real simulado de una microempresa con problemas de inventario (datos de productos, entradas y salidas mal registradas, cantidades inconsistentes).

En equipos pequeños, los aprendices analizan la situación y reflexionan sobre las causas del problema y posibles soluciones desde el punto de vista lógico y algorítmico.

Los aprendices identifican:

Errores lógicos en los registros.

Decisiones mal tomadas por falta de validaciones.

Procesos que podrían automatizarse.

El equipo propone, de forma oral o escrita:

Qué reglas lógicas deberían existir.

Qué datos se deben controlar.

Qué pasaría si no se usan algoritmos ni estructuras de control.

Ambiente requerido: Aula de clase

Estrategias o técnicas didácticas activas: Lluvia de ideas colaborativa y análisis de casos

Materiales de formación: Computador, televisor, hojas, lapiz

Material de apoyo: ANEXO 1 Video, imagenes... de tres casos breves de eventos deportivos reales con fallos organizativos

Duración de la actividad: 4 horas.

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

Descripción de la actividad:

Los aprendices trabajan en equipos para analizar un caso real de inventario de una microempresa con registros inconsistentes. Identifican errores en entradas y salidas de productos, detectan qué datos deben validarse y proponen reglas lógicas y posibles soluciones usando algoritmos. Finalmente, comparten sus conclusiones con el grupo

para reflexionar sobre cómo la programación puede resolver problemas reales de inventario.

3.2.1 . Contexto del entorno productivo real

Entorno:

En una microempresa comercial, como una papelería o ferretería ubicada en un barrio urbano, se presenta un problema constante en la gestión del inventario. El encargado debe registrar, actualizar y controlar las entradas y salidas de productos para garantizar información confiable y una atención eficiente a los clientes. Sin embargo, los registros manuales generan inconsistencias: productos agotados que aparecen disponibles, duplicación de datos y ausencia de alertas cuando el stock es bajo. Este problema surge durante la operación diaria, especialmente en horarios de alta demanda, y se debe a la falta de un sistema automatizado que valide cantidades y registre correctamente las transacciones.

Causas:

Investigación previa revela:

- Registro manual de inventario: Los productos se anotan en hojas o planillas, aumentando la probabilidad de errores humanos.
- Falta de validación de datos: No hay un mecanismo que verifique que las cantidades de salida no superen el stock disponible.
- Duplicación de registros: Se ingresan productos varias veces sin control, generando inconsistencias.
- Ausencia de alertas de stock bajo: No se notifica al encargado cuando las cantidades son insuficientes para la venta.
- Desconocimiento de procedimientos lógicos: El personal no aplica reglas de lógica o algoritmos simples para verificar y organizar la información.

- Limitaciones tecnológicas: No se dispone de un sistema automatizado o software de control de inventario adecuado.
- Alta carga de trabajo en picos de venta: Durante horarios de alta demanda, el registro manual se vuelve más propenso a errores.
- Falta de capacitación en herramientas digitales: El personal no sabe usar correctamente herramientas colaborativas o programas para organizar la información.

3.2.2 Actividad de contextualización e identificación de conocimientos previos

Técnica didáctica activa: Mapa mental colaborativo

Actividad: “Reconociendo la red y sus fallas”

Instrucciones:

- Escuche atentamente la explicación del instructor sobre la situación problema presentada.
- Formar equipos de trabajo de 3 a 4 aprendices.
- Recibir el caso simulado con el listado de productos, cantidades y fechas de caducidad.
- Analizar la información para identificar:
- Productos caducados que siguen disponibles.
- Cantidades duplicadas o incorrectas.
- Registrar los errores encontrados en un documento o hoja de trabajo.
- Proponer reglas lógicas o ideas iniciales para un sistema automatizado que pueda:Detectar productos caducados, alertar sobre bajo inventario, evitar duplicación de registros.
- Organizar las conclusiones en un listado claro y, opcionalmente, un pequeño esquema o diagrama que represente la solución.
- Presentar el entregable al instructor al finalizar la actividad, explicando brevemente el análisis realizado y las ideas de solución.

Entregable : l físico/digital.

Producto: Mapa conceptual grupal que evidencie los conocimientos previos y la comprensión inicial del problema de conectividad.

Ambiente requerido: Aula de clase

Estrategias o técnicas didácticas activas: Mapa mental colaborativo

Materiales de formación: Papel periódico, marcadores, equipos de computo.

Material de apoyo: ANEXO 2

Duración de la actividad: 6 horas.

3.3 Actividades de apropiación:

Descripción de la actividad:

3.3.1 Objetivos del aprendizaje

Cognitivos: saber

Comprender los fundamentos de lógica, algoritmos y estructuras de programación necesarios para el desarrollo de soluciones de software.

Procedimentales:hacer

Analizar problemas reales y transformarlos en algoritmos estructurados para su implementación en software.

Valorativos/ actitudinales: ser

Fomentar trabajo colaborativo y comunicación efectiva durante el análisis, diseño e implementación de la solución.

3.3.2 Actividades de aprendizaje por conocimiento del proceso productivo

1. Desarrollo de soluciones de software mediante algoritmos, estructuras de programación y trabajo colaborativo.

Instrucción:

- Analiza los fundamentos de lógica, algoritmos y estructuras de programación a partir de ejemplos prácticos y casos reales de software, identificando cómo se aplican en la resolución de problemas. (Técnica activa: Estudio de caso con trabajo en equipo).
- Identifica las características técnicas de la carga y valida la información con fichas técnicas, asegurando que los datos a utilizar en los algoritmos sean correctos y confiables. (Técnica activa: Revisión guiada y comparación de registros).

Técnica: Esquemas y diagramas de lógica para aplicar en algoritmos. Guiados por el instructor.

2. Normas Tecnicas

Instrucción: Identificar y aplicar normas técnicas en el registro y control de información, garantizando calidad y seguridad en soluciones de software.

ANEXO:

Fichas técnicas de un inventario simulado.

Ejemplos de normas técnicas aplicables al software (documentación SENA, ISO 9001, estándares de programación).

Técnica: Aprendizaje basado en problemas (ABP).

ANEXO TALLER PRINCIPIOS DE ALGORITMO

Agrupación: Equipos colaborativos

3. Herramientas de Validación

Instrucción:

Herramientas de validación de datos

Fichas técnicas: Verifican que los datos de inventario (cantidad, tipo de producto, fecha de caducidad) sean correctos y estén completos.

Listas de chequeo (checklist): Permiten revisar paso a paso que los registros cumplan con normas y procedimientos.
Tablas de control y formatos estandarizados: Comparan la información ingresada con valores esperados o rangos permitidos.

2. **Herramientas de validación de procesos**
Diagramas de flujo: Representan gráficamente los procesos para detectar inconsistencias o pasos faltantes.
Pseudocódigo o pruebas de escritorio: Simulan la ejecución de algoritmos paso a paso para comprobar resultados.
Simuladores o software de prueba (como PSeInt o entornos de programación): Permiten ejecutar los algoritmos y verificar su funcionamiento antes de implementarlos.

3. **Herramientas de validación colaborativa**
Revisión entre pares: Otros miembros del equipo verifican los datos, algoritmos y documentación.
Retroalimentación del instructor: Orienta sobre cumplimiento de normas y procedimientos.
Documentación estandarizada: Fichas de trabajo, diagramas y reportes que permiten comparar resultados con los criterios esperados.
Técnica: Taller práctico con casos simulados.ANEXO matriz legal
Agrupación: Grupos de 2, máximo 3 aprendices.

3.3.3 Tipo de agrupación:

Reflexión inicial: en equipos colaborativos Equipos
Diagnóstico técnico: individual
Propuesta de solución: en equipos colaborativos

3.3.4 Productos a elaborar:

1. Informe técnico con:

- Normas técnicas identificadas aplicables al registro de datos, control de inventario y desarrollo de software.
- Errores o inconsistencias detectadas en el caso práctico de inventario o software.
- Propuestas de corrección y aplicación de normas, indicando procedimientos, controles y elementos de validación a implementar.
- Diagramas o esquemas opcionales que muestren la integración de normas en el flujo de trabajo o algoritmos.
- Registro de verificación ambiental y de seguridad, señalando que se cumplieron las condiciones del área de trabajo

2. Video explicativo (3 min) demostrando la solución implementada.

3. Presentación oral para "clientes" (rol-play con docentes).

3.3.5 Temporización (8 sesiones de 2 horas cada una)

Sesión	Actividad	Duración
1	Lluvia de ideas + contexto	2 horas
2	Mapa conceptual y ficha técnica de normas	3 horas
3	Prácticas con software (aplicando lenguaje de programación)	4 horas
4	Diseño de propuesta	3 horas
5	Presentación final	2 horas

Ambiente requerido: Trabajo de Campo

Estrategias o técnicas didácticas activas:

- ✓ Cartelera de ideas
- ✓ Mapa mental
- ✓ Matriz de necesidades

- ✓ Matriz legal
- ✓ Cuadro comparativo
- ✓ Boceto de estructura
- ✓ Documento protocolo
- ✓ Presentación oral o simulación

Materiales de formación: Computador, telefono,.....

Material de apoyo:

1. Matriz de necesidades del usuario
2. Matriz legal de normativas aplicables
3. Protocolo estructurado del evento deportivo escolar (producto final del proyecto

Evidencias de aprendizaje:

- ✓ Matriz de necesidades del usuario
- ✓ Matriz legal de normativas aplicables
- ✓ Protocolo estructurado del evento deportivo escolar (producto final del proyecto

Instrumentos de evaluación:

- ✓ Lista de chequeo de producto
- ✓ Lista de chequeo de desempeño
- ✓ Cuestionario

Duración de la actividad: 40 horas

3.4 Actividades de Transferencia el Conocimiento:

Descripción de la actividad:

3.4.1 Producto, desempeño y conocimiento esperado

Componente	Detalle	Evidencia del resultado de aprendizaje
Cognitivo (Saber)	Comprender los fundamentos de lógica, algoritmos y estructuras de programación aplicados al desarrollo de soluciones de software.	Documento o ficha de análisis donde el aprendiz explique los conceptos de lógica, estructuras y algoritmos aplicados al caso práctico.

Procedimental (Hacer)	Analizar problemas reales y transformarlos en algoritmos estructurados, considerando secuencias, condicionales, ciclos y arreglos; validar información con fichas técnicas; aplicar normas técnicas y procedimientos.	Algoritmos diseñados (pseudocódigo, diagramas de flujo o código en lenguaje de programación) que resuelvan el problema del caso práctico, con registro de validación de datos.
Valorativo / Actitudinal (Ser)	Fomentar trabajo colaborativo y comunicación efectiva durante el análisis, diseño e implementación de la solución; seguir normas técnicas y procedimientos de seguridad y calidad.	Informe de trabajo en equipo, registro de roles, evidencia de coordinación y comunicación, y reporte de cumplimiento de normas y procedimientos.

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

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

Planteamiento de evidencias de aprendizaje

Tipo de evidencia	Descripción	Criterios de evaluación / resultado esperado	Instrumentos y medios de verificación
Evidencia de conocimiento (Cognitivo – Saber)	Documento o ficha técnica donde el aprendiz explique los fundamentos de lógica, algoritmos y estructuras de programación, incluyendo la relación con el caso práctico de inventario.	- Identifica correctamente estructuras secuenciales, condicionales, cíclicas y arreglos. - Comprende normas técnicas y procedimientos aplicables.	Revisión del documento/ficha técnica; lista de cotejo de conceptos; preguntas abiertas de reflexión.
Evidencia de desempeño (Procedimental – Hacer)	Algoritmo o solución de software que resuelva el problema del inventario, aplicando las reglas de validación, cálculo de variables técnicas (como centro de gravedad) y	- Aplica correctamente estructuras algorítmicas y programación según el diseño. - Valida los datos y aplica normas técnicas. - Cumple con condiciones de	Observación directa del desarrollo del algoritmo; prueba de escritorio; revisión de

	asegurando la integridad de los datos según fichas técnicas y normas.	seguridad y ambientales en la simulación práctica.	código o pseudocódigo; checklist de pasos y validaciones.
Evidencia de producto	Ficha técnica final o documento que integre: - Normas técnicas aplicadas - Errores detectados y propuestas de solución - Algoritmo final - Registro de validación de datos y cumplimiento de condiciones de seguridad	- Documento completo y coherente que refleje análisis, aplicación de normas, solución de problemas y validación de información.	Revisión del entregable; rúbrica de evaluación de contenido, claridad y aplicabilidad; esquema o diagrama opcional validado.
Evidencia actitudinal / valorativa (Valorativo – Ser)	Informe de trabajo en equipo y registro de roles, donde se evidencie comunicación efectiva, coordinación, responsabilidad y aplicación de normas y procedimientos durante el desarrollo del software.	- Participa activamente en el equipo y mantiene comunicación clara y efectiva. - Demuestra responsabilidad y compromiso con la calidad de la solución. - Sigue normas y procedimientos en todo momento.	Observación directa durante la actividad; registro de roles y actitudes; rúbrica de participación y colaboración.

Plan de Evaluación

Tipo de evaluación	Objeto de evaluación	Actividad / Instrumento	Criterios de éxito	Momento de aplicación
Formativa	Conocimiento (Cognitivo)	- Lectura y análisis de fundamentos de lógica, algoritmos y estructuras de programación. - Identificación de normas técnicas aplicables.	- Explica correctamente los conceptos de lógica, estructuras y algoritmos. - Reconoce normas técnicas aplicables al software y al	Durante las primeras sesiones, antes de implementar la solución.

			control de inventario.	
Formativa	Procedimiento (Hacer)	- Resolución de problemas parciales con pseudocódigo, diagramas de flujo o algoritmos simples. - Validación de datos usando fichas técnicas y listas de control.	- Aplica estructuras secuenciales, condicionales y cíclicas de manera correcta. - Identifica y corrige errores en datos y registros.	Durante la práctica de análisis y diseño, previo al producto final.
Formativa	Actitud / Valor (Ser)	- Trabajo colaborativo en equipo. - Comunicación efectiva y cumplimiento de normas de seguridad y procedimientos .	- Participa activamente y mantiene comunicación clara. - Aplica normas, procedimientos y medidas de seguridad.	Durante todo el proceso formativo, mediante observación directa.
Sumativa	Producto final	- Ficha técnica final o producto pedido.		

5. Criterios de Éxito

- **Mínimo aprobatorio:** 70/100 puntos (sumando formativa y sumativa).
- **Retroalimentación:** Sesión grupal post-evaluación para analizar aciertos/mejoras.

Nota: Este plan integra lo **técnico** (normas, herramientas),
lo **procedimental** (diagnóstico, solución) y lo **actitudinal** (trabajo en equipo, puntualidad).
¿Necesitas ajustar ponderaciones o agregar otro instrumento? 📝🔍

5. GLOSARIO DE TÉRMINOS

Glosario de Términos Técnicos

(Definiciones claras y concisas en contexto de redes y telecomunicaciones)

- **Algoritmo**: Conjunto de instrucciones ordenadas y finitas que permiten resolver un problema o realizar una tarea específica en un sistema.
- **Arreglo (Array)**: Estructura de datos que almacena múltiples valores del mismo tipo en posiciones indexadas, utilizada en programación para organizar información.
- **Condicional**: Estructura de control en programación que permite ejecutar instrucciones solo si se cumple una condición específica.
- **Cíclica (Bucle o Iteración)**: Estructura de control que permite repetir un conjunto de instrucciones hasta que se cumpla una condición determinada.
- **Centro de gravedad**: Punto en un objeto o sistema donde se considera que actúa su peso total; en procesos de carga, permite calcular estabilidad y seguridad.
- **Checklist / Lista de cotejo**: Herramienta de validación que permite verificar paso a paso el cumplimiento de procedimientos o normas.
- **Diagrama de flujo**: Representación gráfica de un algoritmo o proceso, que muestra pasos, decisiones y secuencias.
- **Fichas técnicas**: Documentos que registran características detalladas de productos, cargas o información de inventario, utilizados para la validación de datos.
- **Pseudocódigo**: Representación de un algoritmo en lenguaje estructurado, cercano al lenguaje humano, que permite planificar la solución antes de programarla.
- **Secuencial**: Estructura de programación donde las instrucciones se ejecutan en un orden lineal, de principio a fin.
- **Validación de datos**: Proceso de verificar que la información ingresada sea correcta, completa y consistente según normas y fichas técnicas.

6. REFERENTES BILBIOGRÁFICOS

- Sommerville, I. (2016). Ingeniería de software (10.ª ed.). Pearson Educación.
- Fuente sobre metodologías de desarrollo de software, diseño y documentación de sistemas.
- Pressman, R. S., & Maxim, B. R. (2019). Ingeniería de software: Un enfoque práctico (9.ª ed.). McGraw-Hill.

- Soporte para procesos de análisis, diseño y pruebas de software.
- Larman, C. (2018). Aplicando UML y patrones: Una introducción a la ingeniería orientada a objetos y diseño ágil. Pearson Educación.
- Base para documentación técnica y representación de algoritmos mediante diagramas de flujo y pseudocódigo.

ISO 9001:2015. Sistemas de gestión de calidad – Requisitos.

Normas de calidad aplicables a procesos y documentación de software.

SENA (Servicio Nacional de Aprendizaje). (2020). Guía de competencias laborales y formación por proyectos. Bogotá, Colombia.Fuente oficial sobre enfoque por competencias, ABP y planificación de evidencias de aprendizaje.

McKinney, W. (2017). Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython (2.ª ed.). O'Reilly Media.

Base práctica para validación de datos y programación aplicada a inventarios y procesos de software.

Lafore, R. (2018). Estructuras de datos y algoritmos en Java (2.ª ed.). Pearson Educación. Soporte para algoritmos, estructuras condicionales, secuenciales y cíclicas.

Pineda, J., & Hernández, A. (2019). Técnicas de trabajo colaborativo y aprendizaje basado en proyectos. Editorial Académica Española.Orientación pedagógica para ABP, trabajo en equipo y evaluación formativa y sumativa.

7. CONTROL DEL DOCUMENTO

	Nombre	Cargo	Dependencia	Fecha
Autor (es)	**JHOAN DAVID NIÑO RIVEROS**	**Instructor**	**CIMM**	**06/02/2026**

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

	Nombre	Cargo	Dependencia	Fecha	Razón del Cambio
Autor (es)					