<h1>Del código al mundo real: anatomía, evolución y elección de los lenguajes de programación</h1>

Ingeniería | Ingeniería de sistemas | Aprendizaje Basado en Problemas

Descripción

Este plan de clase introduce a los estudiantes de Ingeniería de Sistemas en el concepto de programa de computación, su evolución histórica y las principales formas de clasificar los lenguajes de programación. Mediante la metodología de Aprendizaje Basado en Problemas, los estudiantes resolverán un caso relacionado con el diseño de una solución tecnológica para una organización que debe seleccionar lenguajes y herramientas de desarrollo.

Durante la sesión, analizarán cómo una instrucción se transforma desde el lenguaje máquina hasta lenguajes de alto nivel como FORTRAN, COBOL, BASIC, C, C++ y Java. También diferenciarán el funcionamiento de intérpretes y compiladores, y clasificarán lenguajes según sus paradigmas: estructurados, imperativos, funcionales, lógicos y orientados a objetos.

La relevancia del tema se relaciona con decisiones reales de la ingeniería de sistemas. Elegir un lenguaje no depende únicamente de su popularidad, sino de factores como el problema, el rendimiento, la mantenibilidad, el dominio de aplicación y la disponibilidad de herramientas. Los estudiantes conectarán estos conceptos con aplicaciones cotidianas como sistemas bancarios, videojuegos, aplicaciones móviles, inteligencia artificial y plataformas web. Al finalizar, podrán argumentar técnicamente una selección de lenguajes para un problema específico y reconocer que los programas son modelos formales de procesos, expresados mediante instrucciones que el computador puede ejecutar.

Objetivos de Aprendizaje

- **Explicar** qué es un programa de computación y relacionar su evolución con los principales hitos históricos de los lenguajes de programación.
- **Comparar** el funcionamiento de un intérprete y un compilador, identificando ventajas, limitaciones y contextos de uso.
- **Clasificar** lenguajes de programación según los paradigmas estructurado, imperativo, funcional, lógico y orientado a objetos.
- **Argumentar** la selección de lenguajes de programación para resolver un problema de ingeniería de sistemas, considerando sus características técnicas y el contexto de aplicación.

Recursos Necesarios

- Proyector, computador del docente y conexión a Internet.

- Presentación breve con una línea de tiempo de los lenguajes: lenguaje máquina, ensamblador, FORTRAN, COBOL, BASIC, C, C++ y Java.
- Video introductorio de 3 a 4 minutos sobre la evolución de los lenguajes de programación.
- Un tablero físico o digital, como Microsoft Whiteboard, Miro o Jamboard.
- Una hoja impresa por grupo con el caso problema y las tarjetas de análisis.
- Tarjetas impresas con nombres, características y ejemplos de lenguajes de programación.
- Computadores o teléfonos con acceso a Internet, uno por grupo de 3 o 4 estudiantes.
- Editor o compilador en línea, como Programiz, OnlineGDB o Replit.
- Fragmentos breves de código en lenguaje máquina simbólico, ensamblador, C, Java, Prolog y Haskell, proporcionados por el docente.
- Lista de cotejo para la clasificación y rúbrica breve para la argumentación final.

Requisitos Previos

- Reconocer qué es un algoritmo y diferenciarlo de un programa.
- Identificar datos de entrada, procesos y resultados en una situación problemática.
- Comprender nociones básicas de hardware, memoria y unidad central de procesamiento.
- Interpretar estructuras simples de pseudocódigo, como secuencia, decisión y repetición.
- Utilizar un navegador web y trabajar colaborativamente con documentos digitales.
- Haber abordado previamente la resolución básica de problemas mediante algoritmos o pseudocódigo.

Actividades

Fase de Inicio

Tiempo estimado: 20 minutos.

Propósito de la sesión: Activar conocimientos previos y plantear un problema auténtico que permita comprender qué es un programa, cómo evolucionaron los lenguajes y por qué la selección del lenguaje es una decisión de ingeniería.

Activación de conocimientos previos: “¿Qué ocurre cuando presiono una tecla?”

Tiempo: 8 minutos.

- **Docente:** Proyecta la pregunta: “Cuando escribo una contraseña y presiono Enter, ¿qué instrucciones debe ejecutar el computador para validar el acceso?”.
- **Estudiantes:** Responden individualmente en una tarjeta o formulario digital, indicando entradas, procesos y salida.
- **Docente:** Solicita a tres estudiantes compartir sus respuestas y pregunta: “¿La intención humana se convierte directamente en instrucciones ejecutables?”.

- **Estudiantes:** Comparan sus respuestas con las de sus compañeros e identifican la necesidad de traducir las instrucciones a un lenguaje comprensible para la máquina.

El docente recoge ideas clave en el tablero: algoritmo, instrucción, datos, procesamiento, resultado y programa.

Motivación y enganche: de los unos y ceros a las aplicaciones actuales

Tiempo: 7 minutos.

- **Docente:** Presenta un fragmento visual de instrucciones binarias y pregunta: “¿Podría un equipo de desarrollo moderno mantener una aplicación bancaria escribiendo únicamente 0 y 1?”.
- **Docente:** Muestra un video breve sobre la evolución de los lenguajes y, al terminar, plantea: “¿Qué problema intentó resolver cada nueva generación de lenguajes?”.
- **Estudiantes:** Anotan un cambio observado y lo relacionan con legibilidad, productividad, portabilidad o control del hardware.

Contextualización y presentación del reto

Tiempo: 5 minutos.

Docente: Presenta el reto: “Una universidad desea modernizar su sistema de matrículas, conservar información histórica, ofrecer una aplicación móvil y crear un módulo inteligente para detectar conflictos de horarios. El equipo debe recomendar lenguajes y justificar la elección”.

Estudiantes: Formulan una primera hipótesis sobre qué lenguajes utilizarían y qué información necesitarían para decidir. El docente aclara que la respuesta se construirá durante la sesión y organiza grupos de 3 o 4 integrantes.

Fase de Desarrollo

Tiempo estimado: 80 minutos.

Presentación del contenido mediante ABP: El docente no inicia con una exposición extensa. Entrega el caso, solicita que los grupos identifiquen lo que saben y lo que necesitan investigar, y utiliza las preguntas de los estudiantes para introducir los conceptos. La explicación se construye con ejemplos breves: un programa es un conjunto organizado de instrucciones y datos que especifica a un computador cómo transformar entradas en resultados; el lenguaje máquina representa instrucciones directamente ejecutables por el procesador; el ensamblador utiliza mnemónicos; FORTRAN favoreció el cálculo científico; COBOL se orientó a aplicaciones de negocios; BASIC facilitó el aprendizaje; C aportó eficiencia y portabilidad; C++ incorporó abstracción y orientación a objetos; y Java promovió portabilidad mediante la máquina virtual.

Actividad 1: Línea de tiempo argumentada

Tiempo: 25 minutos. **Contribuye a los objetivos 1 y 4.**

- **Docente:** Entrega a cada grupo tarjetas con los lenguajes y descripciones históricas mezcladas. Indica: “Ordenen los elementos y relacionen cada lenguaje con la necesidad tecnológica que ayudó a resolver”.
- **Estudiantes:** Construyen una línea de tiempo desde lenguaje máquina hasta Java.

- **Estudiantes:** Incorporan una frase para cada hito: propósito, nivel de abstracción y posible área de uso.
- **Docente:** Formula preguntas guía: “¿Qué ventaja ofrece un lenguaje de alto nivel frente al lenguaje máquina?”, “¿Por qué C sigue siendo relevante?”, “¿Qué diferencia histórica observan entre FORTRAN, COBOL y BASIC?”.
- **Estudiantes:** Comparan la línea de tiempo con otra pareja y corrigen dos posibles inconsistencias.

Evidencia: Línea de tiempo con ocho hitos y una justificación de la evolución.

Actividad 2: ¿Compilar o interpretar?

Tiempo: 25 minutos. **Contribuye a los objetivos 2 y 4.**

- **Docente:** Presenta el mismo algoritmo, calcular el promedio de tres notas, en pseudocódigo y en un fragmento sencillo de C y JavaScript.
- **Docente:** Explica mediante un diagrama que el compilador traduce el programa antes de ejecutarlo, mientras que el intérprete analiza y ejecuta instrucciones durante la ejecución, aclarando que los entornos actuales pueden combinar técnicas.
- **Estudiantes:** Ejecutan, cuando sea posible, los ejemplos en OnlineGDB, Programiz o Replit.
- **Estudiantes:** Completan una tabla con las categorías: momento de traducción, velocidad de ejecución, portabilidad, detección de errores y ejemplos.
- **Docente:** Pregunta: “¿Qué preferirían para detectar rápidamente cambios durante el desarrollo?”, “¿Qué considerarían en un sistema donde el rendimiento sea crítico?”, “¿Es correcto afirmar que un lenguaje es siempre exclusivamente compilado o interpretado?”.
- **Estudiantes:** Elaboran una conclusión de cuatro líneas sobre la decisión para el sistema de matrículas.

Evidencia: Tabla comparativa y recomendación técnica preliminar.

Actividad 3: Mapa de paradigmas y decisión de arquitectura

Tiempo: 15 minutos. **Contribuye a los objetivos 3 y 4.**

- **Docente:** Distribuye tarjetas con ejemplos: C, Pascal, Java, C++, Haskell, Lisp, Prolog y Python. Indica: “Clasifiquen cada ejemplo según el paradigma predominante y escriban una característica observable”.
- **Estudiantes:** Organizan las tarjetas en estructurados, imperativos, funcionales, lógicos y orientados a objetos. Pueden ubicar un lenguaje en más de una categoría si justifican su decisión.
- **Docente:** Aclara que las categorías pueden superponerse: por ejemplo, Java es imperativo y orientado a objetos; C es imperativo y estructurado; Haskell es funcional; Prolog es lógico.
- **Estudiantes:** Seleccionan dos lenguajes para el caso problema y escriben una razón técnica para cada uno.

Evidencia: Mapa de clasificación y dos decisiones justificadas.

Diferenciación y transición

Para quienes necesiten apoyo, el docente proporciona un glosario visual, ejemplos parcialmente resueltos y una tabla con palabras clave como “objetos”, “reglas”, “funciones” e “instrucciones”. También permite explicar oralmente la clasificación antes de escribirla. Quienes terminan antes deben analizar Python y responder si puede pertenecer a más

de un paradigma, justificando su respuesta con dos características.

Al finalizar la Actividad 1, el docente indica: “Ya sabemos por qué surgieron distintos lenguajes; ahora verificaremos cómo se convierten en instrucciones ejecutables”. Después de la Actividad 2, enlaza con la clasificación: “La forma de traducir un programa no es lo mismo que la forma de pensar y organizar la solución; por eso analizaremos los paradigmas”.

Fase de Cierre

Tiempo estimado: 20 minutos.

Síntesis colectiva

Tiempo: 8 minutos.

Docente: Dibuja cinco recuadros en el tablero: programa, evolución histórica, compilador, intérprete y paradigmas. Cada grupo aporta una idea esencial y un ejemplo. El docente organiza las respuestas en un mapa conceptual que conecte “problema”, “algoritmo”, “programa”, “lenguaje” y “ejecución”.

Estudiantes: Entregan una síntesis de tres ideas: qué es un programa, cuál fue el principal cambio histórico observado y cómo se elegiría un lenguaje para un contexto específico.

Reflexión metacognitiva y retroalimentación

Tiempo: 7 minutos.

Los estudiantes responden individualmente las siguientes preguntas:

- “¿Cómo puedo explicar, con mis propias palabras, la diferencia entre un algoritmo y un programa?”
- “¿Qué evidencia me permite decidir si conviene un proceso compilado, interpretado o híbrido?”
- “¿Qué paradigma elegiría para el sistema de matrículas y qué característica respalda mi decisión?”

Docente: Revisa respuestas representativas, corrige confusiones entre lenguaje y paradigma, y ofrece retroalimentación inmediata utilizando la lista de cotejo.

Transferencia y reto

Tiempo: 5 minutos.

Docente: Relaciona lo aprendido con aplicaciones futuras: desarrollo web, sistemas embebidos, análisis de datos, inteligencia artificial y sistemas empresariales. Asigna el siguiente reto: seleccionar una aplicación cotidiana, identificar dos lenguajes que podrían participar en su construcción, clasificarlos por paradigma y explicar si su ejecución dependería principalmente de compilación, interpretación o una combinación de ambas.

Estudiantes: Registran el reto en su portafolio y formulan una pregunta que deseen investigar en la próxima clase.

Evaluación

Estrategia de evaluación

Se aplicará evaluación **diagnóstica, formativa y sumativa** durante la única sesión.

Evaluación diagnóstica: Fase de Inicio, minutos 1 a 8

Se revisan las respuestas a la pregunta sobre la validación de una contraseña. Se identifican conocimientos previos sobre algoritmos, instrucciones, entradas, procesos y resultados. Esta información orienta las aclaraciones del docente, sin asignar calificación.

Evaluación formativa: Fase de Desarrollo, minutos 20 a 100

- **Criterio vinculado al objetivo 1:** Explica el concepto de programa y relaciona correctamente los principales hitos históricos con las necesidades que resolvieron.
- **Criterio vinculado al objetivo 2:** Compara intérpretes y compiladores considerando traducción, ejecución, errores, portabilidad y rendimiento.
- **Criterio vinculado al objetivo 3:** Clasifica ejemplos de lenguajes según los paradigmas y reconoce que algunos lenguajes pueden ser multiparadigma.
- **Criterio vinculado al objetivo 4:** Fundamenta la selección de lenguajes para el sistema de matrículas con argumentos técnicos pertinentes.

Instrumentos: lista de cotejo para la línea de tiempo, tabla de compilación e interpretación, observación directa, preguntas de andamiaje y coevaluación entre grupos.

Evaluación sumativa: Fase de Cierre, minutos 100 a 120

La evidencia principal será la recomendación grupal para el caso problema, complementada con la síntesis individual y el ticket de reflexión. Se utilizará una rúbrica breve con cuatro criterios: exactitud conceptual, integración de la historia, clasificación de paradigmas y calidad de la argumentación. Cada criterio se valorará en cuatro niveles: inicial, básico, competente y sobresaliente.

Evidencias de aprendizaje

- Línea de tiempo argumentada con lenguaje máquina, ensamblador, FORTRAN, COBOL, BASIC, C, C++ y Java.
- Tabla comparativa entre intérpretes y compiladores y recomendación aplicada al caso.
- Mapa de clasificación de lenguajes estructurados, imperativos, funcionales, lógicos y orientados a objetos.
- Recomendación técnica grupal para el sistema universitario de matrículas.
- Respuestas individuales de síntesis y reflexión metacognitiva.