Fundamentos y Clasificación de Lenguajes de Programación

Ingeniería | Ingeniería de sistemas | para estudiantes de educación técnica/tecnológica | 4 semanas

Descripción del Curso

Este curso ofrece una introducción integral a los lenguajes de programación, enfocándose en su clasificación, características y aplicaciones dentro del ámbito de la ingeniería de sistemas. Está diseñado para estudiantes de educación técnica y tecnológica que buscan comprender los fundamentos que sustentan los diversos lenguajes utilizados en la industria del software.

A lo largo de cuatro semanas, los estudiantes explorarán desde conceptos básicos hasta la identificación de diferentes tipos de lenguajes de programación, reconociendo sus ventajas y limitaciones. El enfoque metodológico combina exposiciones teóricas con actividades prácticas y análisis de casos para facilitar la comprensión y aplicación de los contenidos.

Al finalizar el curso, los estudiantes serán capaces de clasificar distintos lenguajes de programación según su paradigma y nivel de abstracción, identificar sus usos adecuados y seleccionar el lenguaje más pertinente para resolver problemas específicos en proyectos de ingeniería de sistemas.

Objetivos Generales

- Describir los conceptos fundamentales de los lenguajes de programación y su evolución histórica.
- Clasificar los lenguajes de programación según su nivel y paradigma de programación.
- Comparar diferentes lenguajes de programación en función de sus características y aplicaciones.
- Analizar casos prácticos para seleccionar el lenguaje más adecuado según el tipo de proyecto.
- Implementar ejemplos sencillos que demuestren el uso básico de distintos lenguajes de programación.

Competencias

- Clasificar los lenguajes de programación según sus paradigmas y niveles de abstracción.
- Identificar las características principales de los lenguajes de programación más comunes en la industria.
- Analizar ventajas y limitaciones de diferentes tipos de lenguajes para aplicaciones específicas.
- Seleccionar el lenguaje de programación adecuado para resolver problemas técnicos concretos.
- Aplicar conceptos fundamentales de los lenguajes de programación en contextos prácticos.

Requerimientos

- Conocimientos básicos de informática y lógica de programación.
- Acceso a computador con software básico de programación o editores de texto.
- Material bibliográfico o acceso a recursos digitales sobre fundamentos de programación.
- Habilidades básicas en el manejo de sistemas operativos y navegación web.

Unidades del Curso

Unidad 1: Introducción a los Lenguajes de Programación

Objetivos de Aprendizaje

- Al finalizar la unidad, el estudiante será capaz de definir los conceptos básicos de un lenguaje de programación y su función en la ingeniería de sistemas, utilizando ejemplos simples para ilustrar su importancia.
- Al finalizar la unidad, el estudiante será capaz de describir la evolución histórica de los lenguajes de programación, identificando los hitos principales y su impacto en el desarrollo tecnológico.
- Al finalizar la unidad, el estudiante será capaz de enumerar y clasificar los principales tipos de lenguajes de programación según su nivel y paradigma, justificando brevemente las diferencias entre ellos.
- Al finalizar la unidad, el estudiante será capaz de explicar la relevancia de los lenguajes de programación en proyectos de ingeniería de sistemas, mediante la presentación de casos prácticos básicos.

Contenidos Temáticos

1. Conceptos básicos de los lenguajes de programación

- **Definición de lenguaje de programación:** Explicación clara y sencilla de qué es un lenguaje de programación, destacando su función como medio para que los humanos puedan comunicarse con las computadoras.
- **Función en la ingeniería de sistemas:** Cómo los lenguajes de programación permiten diseñar, desarrollar, mantener y optimizar sistemas de software.
- **Ejemplos simples:** Presentación de ejemplos básicos de lenguajes (por ejemplo, instrucciones en pseudocódigo y fragmentos simples en Python o JavaScript) para ilustrar la importancia y utilidad de estos lenguajes en la resolución de problemas.

2. Historia y evolución de los lenguajes de programación

- **Primeras generaciones de lenguajes:** Desde el lenguaje máquina y código binario hasta los lenguajes ensambladores.
- **Lenguajes de alto nivel:** Aparición de lenguajes como FORTRAN, COBOL y BASIC, y su impacto en la accesibilidad y eficiencia del desarrollo de software.
- **Evolución hacia paradigmas modernos:** Introducción a lenguajes estructurados, orientados a objetos, funcionales y scripting, destacando hitos clave como C, Pascal, Java y Python.

- **Impacto tecnológico:** Cómo la evolución de los lenguajes ha influido en el avance de la tecnología y la ingeniería de sistemas.

3. Clasificación de los lenguajes de programación

- **Según el nivel:**
  - Lenguajes de bajo nivel (lenguaje máquina y ensamblador).
  - Lenguajes de alto nivel (C, Java, Python, etc.).
- **Según el paradigma de programación:**
  - Imperativo (ejemplo: C, Pascal).
  - Orientado a objetos (ejemplo: Java, C++).
  - Funcional (ejemplo: Haskell, Lisp).
  - Declarativo (ejemplo: SQL, Prolog).
  - Scripting (ejemplo: JavaScript, Python).
- **Diferencias y características:** Breve explicación de las diferencias entre los niveles y paradigmas, con ejemplos que permitan justificar su uso y características principales.

4. Relevancia de los lenguajes de programación en proyectos de ingeniería de sistemas

- **Importancia en desarrollo de software:** Rol central de los lenguajes en la creación de soluciones tecnológicas eficientes y escalables.
- **Casos prácticos básicos:** Presentación de ejemplos reales o simulados de proyectos sencillos donde la elección del lenguaje impacte en el resultado (como desarrollo de aplicaciones web, automatización de procesos o sistemas embebidos).
- **Consideraciones para la elección de un lenguaje:** Factores como la facilidad de uso, rendimiento, comunidad y soporte.

Actividades

Actividad 1: Definiendo los lenguajes de programación

Objetivo: Definir los conceptos básicos de un lenguaje de programación y su función en la ingeniería de sistemas.

Descripción:

- El docente presenta una breve exposición sobre qué es un lenguaje de programación y su función.
- Los estudiantes en forma individual escriben una definición propia del concepto y describen una situación en la que un lenguaje de programación sea útil.
- Se comparte en grupos pequeños las definiciones y ejemplos para discutir similitudes y diferencias.
- Finalmente, se realiza una plenaria para recoger las definiciones más claras y completas.

Organización: Individual y grupos pequeños de 3-4 estudiantes.

Producto esperado: Definición escrita individual y lista de ejemplos compartidos en grupo.

Duración estimada: 45 minutos.

Actividad 2: Línea de tiempo de la evolución de los lenguajes de programación

Objetivo: Describir la evolución histórica de los lenguajes de programación y sus hitos principales.

Descripción:

- En grupos, los estudiantes investigan hitos clave en la historia de los lenguajes de programación (máquina, ensamblador, FORTRAN, COBOL, C, Java, Python, etc.).
- Crean una línea de tiempo visual con fechas, nombres y características básicas de cada lenguaje o generación.
- Cada grupo presenta su línea de tiempo y discute el impacto tecnológico de los hitos seleccionados.

Organización: Grupos de 4-5 estudiantes.

Producto esperado: Línea de tiempo gráfica y presentación oral.

Duración estimada: 90 minutos.

Actividad 3: Clasificación y comparación de lenguajes

Objetivo: Enumerar y clasificar los principales tipos de lenguajes de programación y justificar las diferencias.

Descripción:

- El docente entrega una lista de lenguajes variados.
- En parejas, los estudiantes clasifican los lenguajes según nivel (bajo o alto) y paradigma (imperativo, orientado a objetos, funcional, etc.).
- Discuten y escriben una breve justificación para cada clasificación.
- Se realiza una puesta en común para resolver dudas y clarificar conceptos.

Organización: Parejas.

Producto esperado: Tabla de clasificación con justificaciones escritas.

Duración estimada: 60 minutos.

Actividad 4: Análisis de casos prácticos en ingeniería de sistemas

Objetivo: Explicar la relevancia de los lenguajes en proyectos mediante casos prácticos básicos.

Descripción:

- El docente presenta 2 o 3 casos prácticos simples (por ejemplo, desarrollo de una página web, automatización de tareas con scripts, programación de un microcontrolador).
- En grupos, los estudiantes analizan qué tipo de lenguaje sería adecuado para cada caso y justifican su elección considerando las características del proyecto.
- Los grupos exponen sus conclusiones al grupo completo para discutir las diferentes opciones.

Organización: Grupos de 3-4 estudiantes.

Producto esperado: Análisis escrito y presentación oral.

Duración estimada: 75 minutos.

Evaluación

Evaluación diagnóstica

Qué se evalúa: Conocimientos previos sobre qué es un lenguaje de programación y su función básica.

Cómo se evalúa: Cuestionario corto con preguntas abiertas y de opción múltiple al inicio de la unidad.

Instrumento sugerido: Formato digital o en papel con 5 preguntas breves.

Evaluación formativa

Qué se evalúa: Comprensión progresiva de conceptos básicos, evolución histórica, clasificación y aplicación práctica.

- Revisión y retroalimentación de las definiciones individuales y discusiones grupales en Actividad 1.
- Evaluación de la línea de tiempo realizada en Actividad 2 por precisión y claridad.
- Corrección y discusión de la tabla de clasificación y justificaciones en Actividad 3.
- Valoración del análisis y argumentación en casos prácticos de Actividad 4.

Instrumentos sugeridos: Rúbricas para trabajos grupales, listas de cotejo para presentaciones y observación directa.

Evaluación sumativa

Qué se evalúa: Dominio integral de los objetivos de la unidad.

Cómo se evalúa: Examen escrito que incluya:

- Definición y función de los lenguajes de programación con ejemplos.
- Preguntas sobre la evolución histórica y sus hitos.
- Ejercicios de clasificación y justificación de lenguajes.
- Preguntas cortas para explicar la relevancia en proyectos de ingeniería.

Instrumento sugerido: Prueba escrita con preguntas abiertas y de opción múltiple.

Unidad 2: Clasificación por Niveles y Tipos de Lenguajes

Objetivos de Aprendizaje

- Al finalizar la unidad, el estudiante será capaz de identificar y describir las características principales de los lenguajes de bajo, medio y alto nivel mediante ejemplos representativos.
- Al finalizar la unidad, el estudiante será capaz de comparar las diferencias fundamentales entre los niveles de lenguajes de programación evaluando sus ventajas y desventajas en contextos específicos.
- Al finalizar la unidad, el estudiante será capaz de clasificar distintos lenguajes de programación en sus respectivos niveles y tipos basándose en criterios técnicos y usos típicos.

- Al finalizar la unidad, el estudiante será capaz de analizar casos prácticos para seleccionar el nivel de lenguaje de programación más adecuado según los requerimientos del proyecto.
- Al finalizar la unidad, el estudiante será capaz de implementar ejemplos sencillos que demuestren el uso básico de lenguajes de bajo, medio y alto nivel para ilustrar sus diferencias funcionales.

Contenidos Temáticos

1. Introducción a la clasificación de lenguajes de programación

- Concepto de lenguaje de programación y su importancia.
- Motivos para clasificar los lenguajes según niveles y tipos.
- Visión general de la clasificación por niveles: bajo, medio y alto.

2. Lenguajes de bajo nivel

- Definición y características principales.
- Lenguaje máquina: estructura y ejemplos.
- Lenguaje ensamblador: sintaxis, uso y ejemplos representativos.
- Ventajas y desventajas de los lenguajes de bajo nivel.
- Contextos típicos de aplicación (sistemas embebidos, controladores, optimización).

3. Lenguajes de medio nivel

- Concepto y características diferenciadoras respecto a bajo y alto nivel.
- Ejemplos representativos (por ejemplo, C).
- Capacidades de abstracción y control del hardware.
- Ventajas y desventajas en su uso.
- Aplicaciones típicas (desarrollo de sistemas operativos, software de sistema).

4. Lenguajes de alto nivel

- Definición y características clave (abstracción, legibilidad, portabilidad).
- Ejemplos de lenguajes de alto nivel populares (Python, Java, C#, JavaScript).
- Tipos dentro del alto nivel: imperativos, orientados a objetos, funcionales.
- Ventajas y limitaciones en diferentes contextos.
- Usos típicos: desarrollo web, aplicaciones móviles, software empresarial.

5. Comparación entre niveles de lenguajes

- Diferencias fundamentales en sintaxis, abstracción y control del hardware.
- Comparación de ventajas y desventajas para cada nivel.
- Impacto en la eficiencia, facilidad de desarrollo y mantenimiento.

6. Clasificación práctica de lenguajes según criterios técnicos y de uso

- Criterios para la clasificación: nivel de abstracción, propósito, paradigma.
- Ejemplos concretos y su clasificación.
- Ejercicios de clasificación de lenguajes dados.

7. Selección del lenguaje adecuado según requerimientos del proyecto

- Factores a considerar: rendimiento, portabilidad, facilidad de uso, soporte.
- Análisis de casos prácticos.
- Justificación técnica de la elección del nivel de lenguaje.

8. Implementación de ejemplos básicos en lenguajes de diferentes niveles

- Ejemplo sencillo en lenguaje de bajo nivel (ensamblador básico o pseudocódigo de máquina).
- Ejemplo en lenguaje de medio nivel (por ejemplo, programa simple en C).
- Ejemplo en lenguaje de alto nivel (programa sencillo en Python o Java).
- Análisis comparativo de los ejemplos para ilustrar las diferencias.

Actividades

Actividad 1: Identificación y descripción de características de lenguajes

Objetivo: Identificar y describir características de lenguajes de bajo, medio y alto nivel.

Descripción:

- Se entrega a los estudiantes una lista de lenguajes (ensamblador, C, Python, entre otros).
- En parejas, investigan y describen para cada lenguaje sus características principales.
- Presentan un cuadro comparativo con ejemplos representativos.

Organización: Parejas

Producto esperado: Cuadro comparativo con descripciones y ejemplos.

Duración: 60 minutos

Actividad 2: Debate sobre ventajas y desventajas de niveles de lenguajes

Objetivo: Comparar diferencias fundamentales y evaluar ventajas y desventajas.

Descripción:

- Se forman dos grupos, cada uno defiende el uso de un nivel de lenguaje (bajo o alto).
- Se discuten ventajas y desventajas en distintos contextos (por ejemplo, eficiencia vs facilidad de desarrollo).
- Finalmente, se realiza una plenaria para consensuar criterios.

Organización: Grupos

Producto esperado: Listado argumentado de ventajas y desventajas por nivel.

Duración: 45 minutos

Actividad 3: Clasificación práctica de lenguajes y análisis de casos

Objetivo: Clasificar lenguajes según criterios técnicos y analizar casos para seleccionar el nivel adecuado.

Descripción:

- Se presentan varios lenguajes y se pide a los estudiantes clasificarlos en bajo, medio o alto nivel.
- Se entregan casos prácticos de proyectos con requerimientos específicos.
- En grupos, analizan y deciden qué nivel de lenguaje es más adecuado, justificando la elección.

Organización: Grupos

Producto esperado: Informe breve con clasificación y justificación para cada caso.

Duración: 75 minutos

Actividad 4: Implementación de ejemplos básicos en distintos niveles

Objetivo: Implementar ejemplos sencillos para ilustrar diferencias funcionales entre niveles.

Descripción:

- Individualmente, los estudiantes programan un ejemplo básico en un lenguaje de alto nivel (por ejemplo, imprimir un mensaje).
- En parejas, revisan y analizan un ejemplo proporcionado de lenguaje de medio nivel (C).
- En conjunto, se revisa un ejemplo simple en lenguaje de bajo nivel (ensamblador o código máquina) proporcionado por el docente.
- Discusión grupal sobre las diferencias observadas en sintaxis, complejidad y aplicación.

Organización: Individual y en parejas

Producto esperado: Código fuente de ejemplos y análisis escrito de diferencias.

Duración: 90 minutos

Evaluación

Evaluación diagnóstica

Qué se evalúa: Conocimientos previos sobre tipos y niveles de lenguajes de programación.

Cómo se evalúa: Cuestionario corto de opción múltiple y preguntas abiertas.

Instrumento sugerido: Formulario en papel o digital con 10 preguntas breves.

Evaluación formativa

Qué se evalúa: Progreso en la identificación, comparación y clasificación de lenguajes durante las actividades.

Cómo se evalúa: Observación directa, revisión de productos parciales (cuadros comparativos, debates, informes).

Instrumento sugerido: Rúbrica para evaluar participación, calidad del análisis y justificación en actividades grupales.

Evaluación sumativa

Qué se evalúa: Competencia para identificar, comparar, clasificar y seleccionar lenguajes así como implementar ejemplos básicos.

Cómo se evalúa: Examen escrito con preguntas de desarrollo y ejercicios prácticos de clasificación y análisis de casos; entrega de código fuente con ejemplos básicos comentados.

Instrumento sugerido: Prueba escrita y portafolio digital de ejemplos implementados.

Unidad 3: Paradigmas de Programación

Objetivos de Aprendizaje

- Al finalizar la unidad, el estudiante será capaz de identificar los principales paradigmas de programación, como imperativo, declarativo, orientado a objetos y funcional, mediante la revisión de ejemplos representativos.
- Al finalizar la unidad, el estudiante será capaz de comparar las características y aplicaciones de los paradigmas de programación estudiados, empleando criterios técnicos para diferencias su uso en proyectos específicos.
- Al finalizar la unidad, el estudiante será capaz de clasificar lenguajes de programación según su paradigma predominante, utilizando análisis de sus estructuras y sintaxis básicas.
- Al finalizar la unidad, el estudiante será capaz de implementar ejemplos sencillos que demuestren el uso básico de distintos paradigmas de programación, aplicando estructuras y conceptos fundamentales de cada uno.
- Al finalizar la unidad, el estudiante será capaz de analizar casos prácticos para seleccionar el paradigma de programación más adecuado, justificando su elección con base en las características del proyecto.

Contenidos Temáticos

1. Introducción a los Paradigmas de Programación

- Definición de paradigma de programación: concepto y importancia en el desarrollo de software.
- Historia y evolución de los paradigmas de programación.
- Relación entre paradigmas y lenguajes de programación.

2. Paradigma Imperativo

- Concepto y características principales: instrucciones secuenciales, uso de variables y estado.
- Estructuras básicas: asignación, condicionales, bucles.
- Ejemplos representativos: lenguajes como C, Pascal.
- Ventajas y desventajas del paradigma imperativo.
- Aplicaciones típicas en proyectos reales.

3. Paradigma Declarativo

- Concepto y características: enfoque en el qué se desea lograr, no en el cómo.

- Subparadigmas principales:
  - Programación lógica (ejemplo: Prolog).
  - Programación funcional (introducción y relación con declarativo).
- Ejemplos representativos y sintaxis básica.
- Ventajas y limitaciones.
- Ámbitos de aplicación comunes.

4. Paradigma Orientado a Objetos (POO)

- Concepto y principios fundamentales: encapsulación, herencia, polimorfismo y abstracción.
- Estructura básica de un programa orientado a objetos: clases, objetos, métodos y atributos.
- Ejemplos representativos: lenguajes como Java, C++ y Python.
- Ventajas del paradigma orientado a objetos en la gestión de proyectos complejos.
- Aplicaciones típicas y ejemplos prácticos.

5. Paradigma Funcional

- Concepto y características: funciones puras, inmutabilidad, funciones de orden superior.
- Diferencias con paradigmas imperativo y orientado a objetos.
- Ejemplos representativos: lenguajes como Haskell, Lisp, y uso funcional en lenguajes multiparadigma.
- Ventajas para la programación concurrente y paralela.
- Ámbitos de aplicación y casos prácticos.

6. Comparación de Paradigmas de Programación

- Análisis de características técnicas: manejo del estado, abstracción, modularidad, legibilidad y mantenibilidad.
- Comparación de aplicaciones: tipos de proyectos y contextos ideales para cada paradigma.
- Ventajas y desventajas relativas.
- Ejemplos comparativos con fragmentos de código simples.

7. Clasificación de Lenguajes de Programación según Paradigma

- Criterios para clasificar lenguajes según su paradigma predominante.
- Análisis de estructuras y sintaxis básicas para identificar el paradigma.
- Lenguajes multiparadigma: características y ejemplos.
- Ejercicios prácticos de clasificación y análisis.

8. Implementación Práctica de Ejemplos Básicos

- Desarrollo de ejemplos sencillos en cada paradigma:
  - Imperativo: programa para cálculo de factorial.

    - Declarativo: consulta simple en Prolog.
    - Orientado a objetos: clase y objeto con método simple.
    - Funcional: función para cálculo recursivo de Fibonacci.
- Explicación detallada de cada ejemplo y sus estructuras.
- Práctica guiada de codificación y ejecución.

9. Análisis y Selección de Paradigmas para Casos Prácticos

- Presentación de casos de estudio reales o simulados.
- Criterios para seleccionar el paradigma más adecuado según características del proyecto.
- Justificación técnica de la elección del paradigma.
- Discusión y reflexión grupal sobre decisiones tomadas.

Actividades

Actividad 1: Identificación y ejemplificación de paradigmas

Objetivo: Contribuir al objetivo de identificar los principales paradigmas de programación mediante la revisión de ejemplos representativos.

Descripción paso a paso:

- El docente presenta fragmentos de código breves que ejemplifican diferentes paradigmas (imperativo, declarativo, orientado a objetos, funcional).
- Los estudiantes analizan en parejas el código, identificando el paradigma al que pertenece cada fragmento y justifican su respuesta.
- Se realiza una puesta en común y discusión guiada para aclarar dudas y consolidar conceptos.

Organización: Parejas

Producto esperado: Lista con la identificación correcta de los paradigmas y justificación breve para cada ejemplo.

Duración estimada: 45 minutos

Actividad 2: Comparación técnica entre paradigmas

Objetivo: Comparar características y aplicaciones de paradigmas empleando criterios técnicos.

Descripción paso a paso:

- Se divide a los estudiantes en grupos y se asigna a cada uno un par de paradigmas para comparar.
- Cada grupo elabora un cuadro comparativo que incluya estructura, manejo del estado, modularidad, ventajas y desventajas, y aplicaciones típicas.
- Los grupos presentan sus cuadros al resto de la clase para retroalimentación y discusión general.

Organización: Grupos de 3-4 estudiantes

Producto esperado: Cuadro comparativo y presentación oral.

Duración estimada: 1 hora 30 minutos

Actividad 3: Clasificación de lenguajes según paradigma

Objetivo: Clasificar lenguajes de programación según su paradigma predominante mediante análisis de estructuras y sintaxis básicas.

Descripción paso a paso:

- El docente provee breves descripciones y fragmentos de código de distintos lenguajes.
- Individualmente, los estudiantes clasifican cada lenguaje en el paradigma adecuado justificando con base en la sintaxis y características observadas.
- Se realiza una revisión grupal para discutir clasificaciones y aclarar conceptos.

Organización: Individual con discusión grupal

Producto esperado: Lista individual con clasificación y justificación.

Duración estimada: 50 minutos

Actividad 4: Implementación de ejemplos básicos en distintos paradigmas

Objetivo: Implementar ejemplos sencillos que demuestren el uso básico de distintos paradigmas.

Descripción paso a paso:

- El docente explica y muestra ejemplos básicos para cada paradigma (factorial imperativo, consulta Prolog, clase y objeto en POO, función recursiva funcional).
- Los estudiantes, en parejas, desarrollan los ejemplos en un entorno de programación adecuado.
- Se verifica el correcto funcionamiento y se discuten las diferencias en la estructura y lógica de cada paradigma.

Organización: Parejas

Producto esperado: Código funcional para cada paradigma y breve informe comparativo.

Duración estimada: 2 horas

Actividad 5: Análisis y selección de paradigma para casos prácticos

Objetivo: Analizar casos prácticos para seleccionar y justificar el paradigma de programación más adecuado.

Descripción paso a paso:

- Se presentan a los estudiantes varios casos prácticos con características específicas del proyecto.
- En grupos, analizan cada caso y seleccionan el paradigma que consideran más adecuado, fundamentando su elección con base en las características técnicas y de aplicación.
- Cada grupo expone su análisis y justificación al resto de la clase para debate y retroalimentación.

Organización: Grupos de 3-4 estudiantes

Producto esperado: Informe escrito y presentación oral del análisis y justificación.

Duración estimada: 1 hora 30 minutos

Evaluación

Evaluación diagnóstica

Qué se evalúa: Conocimientos previos sobre paradigmas de programación y familiaridad con lenguajes de programación.

Cómo se evalúa: Cuestionario breve con preguntas de opción múltiple y de emparejamiento sobre características básicas de paradigmas y lenguajes.

Instrumento sugerido: Cuestionario digital o en papel con 10 preguntas.

Evaluación formativa

Qué se evalúa: Progreso en la identificación, comparación, clasificación e implementación de paradigmas de programación durante las actividades.

Cómo se evalúa: Observación directa del desempeño en actividades prácticas, revisión de productos parciales (cuadros comparativos, códigos, informes), y participación en discusiones.

Instrumento sugerido: Rúbrica para actividades prácticas y lista de cotejo para participación.

Evaluación sumativa

Qué se evalúa: Competencia para identificar, comparar, clasificar, implementar y seleccionar paradigmas de programación de manera integral.

Cómo se evalúa: Examen escrito con preguntas teóricas y análisis de casos, junto con una práctica de programación que incluya implementación y justificación del paradigma usado.

Instrumento sugerido: Prueba escrita y entrega de proyecto práctico con rúbrica de evaluación.

Unidad 4: Aplicaciones y Selección de Lenguajes de Programación

Objetivos de Aprendizaje

- Al finalizar la unidad, el estudiante será capaz de analizar casos prácticos para identificar el lenguaje de programación más adecuado según el tipo de proyecto y sus requisitos específicos.
- Al finalizar la unidad, el estudiante será capaz de comparar ventajas y desventajas de diferentes lenguajes de programación en contextos reales de desarrollo.
- Al finalizar la unidad, el estudiante será capaz de evaluar el impacto del contexto de uso en la selección del lenguaje de programación para aplicaciones específicas.
- Al finalizar la unidad, el estudiante será capaz de justificar la selección de un lenguaje de programación para un proyecto dado, basándose en criterios técnicos y prácticos.
- Al finalizar la unidad, el estudiante será capaz de presentar casos de estudio que demuestren la aplicación efectiva de diferentes lenguajes de programación según el problema planteado.

Contenidos Temáticos

1. Introducción a la Aplicación de Lenguajes de Programación

- Concepto de aplicación práctica de un lenguaje de programación: definición y relevancia.
- Relación entre tipo de proyecto y selección del lenguaje: cómo los requisitos técnicos y funcionales influyen en la elección.

2. Análisis de Casos Prácticos para la Selección de Lenguajes

- Identificación de requisitos del proyecto: funcionales, no funcionales, entorno y recursos.
- Ejemplos de proyectos típicos: desarrollo web, aplicaciones móviles, sistemas embebidos, inteligencia artificial, bases de datos.
- Metodología para analizar casos y seleccionar el lenguaje adecuado.

3. Comparación de Ventajas y Desventajas de Lenguajes de Programación

- Características técnicas relevantes: rendimiento, facilidad de aprendizaje, soporte de comunidad, disponibilidad de librerías, portabilidad.
- Ventajas y desventajas de lenguajes populares en diferentes contextos: Python, Java, C/C++, JavaScript, PHP, Swift, entre otros.
- Impacto de las características del lenguaje en el desarrollo, mantenimiento y escalabilidad del proyecto.

4. Impacto del Contexto de Uso en la Selección del Lenguaje

- Contextos de desarrollo: entornos corporativos, startups, proyectos educativos, desarrollo de software libre, sector público.
- Restricciones del contexto: hardware disponible, plataformas objetivo, requerimientos de tiempo y costo, experiencia del equipo de desarrollo.
- Adaptación del lenguaje y herramientas a las condiciones del entorno de trabajo.

5. Justificación de la Selección del Lenguaje para un Proyecto

- Criterios técnicos: compatibilidad con hardware, rendimiento, seguridad, soporte técnico.
- Criterios prácticos: disponibilidad de desarrolladores, tiempo de desarrollo, costos asociados.
- Construcción de argumentos sólidos para la elección del lenguaje en base a los criterios anteriores.

6. Presentación y Análisis de Casos de Estudio

- Estudio de casos reales donde se aplicaron diferentes lenguajes según el problema.
- Identificación de los factores que motivaron la selección del lenguaje.
- Discusión sobre los resultados obtenidos y lecciones aprendidas.
- Elaboración y presentación de casos de estudio propios por parte de los estudiantes.

Actividades

Actividad 1: Análisis de Caso Práctico para Selección de Lenguaje

Objetivo: Analizar casos prácticos para identificar el lenguaje de programación más adecuado según el tipo de proyecto y sus requisitos específicos.

Descripción:

- Se presenta al grupo un caso práctico detallado (por ejemplo, desarrollo de una aplicación móvil para control de inventarios).
- Los estudiantes, en grupos pequeños, identifican los requisitos funcionales y no funcionales del proyecto.
- Analizan diferentes lenguajes que podrían aplicarse y seleccionan el más adecuado justificando su elección.
- Exponen brevemente sus conclusiones al resto de la clase.

Organización: Grupos de 3-4 estudiantes.

Producto esperado: Informe breve con análisis y justificación de la selección del lenguaje.

Duración estimada: 90 minutos.

Actividad 2: Tabla Comparativa de Lenguajes y Contextos de Uso

Objetivo: Comparar ventajas y desventajas de diferentes lenguajes de programación en contextos reales de desarrollo.

Descripción:

- Cada estudiante elige tres lenguajes de programación distintos.
- Investiga características, ventajas y desventajas de cada uno en un contexto específico (por ejemplo, desarrollo web, sistemas embebidos, IA).
- Elabora una tabla comparativa clara y estructurada.
- Se comparte y discute en plenaria para identificar patrones y diferencias.

Organización: Individual.

Producto esperado: Tabla comparativa con análisis escrito.

Duración estimada: 60 minutos.

Actividad 3: Debate sobre el Impacto del Contexto en la Selección del Lenguaje

Objetivo: Evaluar el impacto del contexto de uso en la selección del lenguaje de programación para aplicaciones específicas.

Descripción:

- Se divide la clase en dos grupos con posturas opuestas respecto a la importancia del contexto (por ejemplo, "El contexto es determinante" vs. "El lenguaje es independiente del contexto").
- Cada grupo prepara argumentos basados en ejemplos reales y criterios técnicos.
- Se realiza un debate moderado donde cada grupo expone y responde a preguntas.
- Finalmente, se reflexiona colectivamente para llegar a conclusiones integradoras.

Organización: Grupos grandes (mitad de la clase cada uno).

Producto esperado: Argumentarios escritos y registro de conclusiones.

Duración estimada: 60 minutos.

Actividad 4: Presentación de Casos de Estudio Propios

Objetivo: Presentar casos de estudio que demuestren la aplicación efectiva de diferentes lenguajes de programación según el problema planteado.

Descripción:

- Los estudiantes seleccionan o investigan un caso real de desarrollo de software.
- Preparan una presentación donde describen el proyecto, el lenguaje seleccionado, razones de la elección, ventajas, desventajas y resultados.
- Presentan ante el grupo y responden preguntas.
- Se realiza retroalimentación grupal.

Organización: Individual o parejas.

Producto esperado: Presentación oral con apoyo visual (diapositivas o similar) y documento de apoyo.

Duración estimada: 2 horas (para preparación y presentación, puede distribuirse en varias sesiones).

Evaluación

Evaluación Diagnóstica

Qué se evalúa: Conocimientos previos sobre lenguajes de programación y criterios básicos para su selección.

Cómo se evalúa: Cuestionario corto con preguntas de opción múltiple y abiertas sobre casos simples de selección de lenguaje.

Instrumento sugerido: Test escrito o digital (plataforma educativa).

Evaluación Formativa

Qué se evalúa: Participación y desempeño en actividades prácticas de análisis, comparación y debate.

Cómo se evalúa: Rúbrica con criterios sobre la calidad del análisis, justificación, argumentación y trabajo en equipo.

Instrumento sugerido: Observación directa, revisión de productos escritos y registros de participación.

Evaluación Sumativa

Qué se evalúa: Capacidad para justificar la selección de lenguajes en proyectos reales y presentar casos de estudio con análisis integral.

Cómo se evalúa: Trabajo final que incluye un estudio de caso completo con análisis y presentación oral o escrita.

Instrumento sugerido: Rúbrica detallada que contemple criterios técnicos, argumentativos y de presentación.