000027

Lenguajes Informáticos 2

INSTITUTOS/S: **INSTITUTO DE TECNOLOGÍA E INGENIERÍA**

CARRERA/S: **Licenciatura en Informática**

RESPONSABLE DE LA ASIGNATURA Y EQUIPO DOCENTE:

Fidel Martínez López
Alan Rodas

AÑO: 2025

CRÉDITOS: 5

CARGA HORARIA DE INTERACCIÓN PEDAGÓGICA: 64

CARGA HORARIA TOTAL: 125

CÓDIGO DE LA MATERIA EN SIU:

000027

UNAHUR
UNIVERSIDAD NACIONAL DE HURLINGHAM

1. Fundamentación

En programación existen diversos paradigmas, lenguajes y herramientas, los cuales deben ser aplicados correctamente a cada problemática en particular. Es necesario entonces cultivar la capacidad de analizar, diseñar y desarrollar soluciones de software robustas, eficientes y mantenibles, para elegir el paradigma y las herramientas más adecuadas para cada problema. Esta materia no solo busca enseñar a programar, sino a pensar como un arquitecto de software, capaz de elegir las herramientas y enfoques adecuados para construir sistemas complejos, eficientes y adaptables a las demandas cambiantes del mundo real. Es una materia que cimienta la versatilidad y el pensamiento crítico indispensables para un informático de alto nivel.

2. Propósitos y/u objetivos

- *Analizar y Comparar Paradigmas de Programación(imperativo, orientado a objetos, funcional, lógico, etc.) identificando sus modelos de cómputo, fortalezas y debilidades.*
- *Caracterizar lenguajes de programación en función de los paradigmas que soportan, sus sistemas de tipos (estáticos, dinámicos, fuertes, débiles), mecanismos de binding (temprano, tardío) y otras características de diseño.*
- *Comparar críticamente diferentes lenguajes de programación y sus capacidades para resolver problemas específicos, justificando su elección.*
- *Aplicar principios de programación funcional para resolver los problemas de manera elegante y eficiente*
- *Diseñar y Utilizar Lenguajes Específicos de Dominio (DSL).*
- *Desarrollar programas basados en especificaciones utilizando técnicas de transformación.*

3. Contenidos mínimos

Estudio de los principales paradigmas de programación, sus modelos de cómputo y estructuras de datos asociadas. Análisis de lenguajes de programación en función de sus características y propósitos. Introducción a los sistemas de tipos y mecanismos de chequeo de tipos, esquemas de binding y variantes del paradigma orientado a objetos, incluyendo técnicas de metaprogramación, reflexión y programación orientada a aspectos. Estudio de la programación funcional: funciones como valores, polimorfismo,

currificación, recursión estructural, e inferencia de tipos. Introducción a la construcción y clasificación de lenguajes específicos de dominio, su integración en aplicaciones multilenguaje y multiparadigma, y el diseño de programas basados en especificaciones mediante técnicas de transformación.

4. Carga horaria

Créditos	*Interacción pedagógica*		*Trabajo Autónomo*	*Total*
5	*Teoría: 32*	*Práctica: 32*	*61*	*125*

4.1 Trabajo autónomo del/la estudiante

Actividad	*Carga horaria*
Lectura de contenidos teóricos apuntes o bibliografía.	*10*
Preparación de evaluaciones.	*20*
Desarrollo de los ejercicios prácticos propuestos.	*31*

5. Programa analítico

UNIDAD 1: *Paradigmas de programación y modelos de cómputo.*

Concepto de paradigma de programación. Modelos de cómputo: imperativo, funcional, orientado a objetos y lógico. Estructuras de datos representativas en los diferentes paradigmas. Clasificación de lenguajes según características y propósito.

UNIDAD 2: *Sistemas de tipos y extensiones del paradigma orientado a objetos.*

Introducción a sistemas de tipos: tipos nominales y estructurales, tipado explícito e implícito. Ducktyping, inferencia de tipos, esquemas de binding (early/late binding). Variantes del paradigma de objetos: bloques, closures, non-local returns. Extensiones: herencia simple y múltiple, mixins, traits, programación orientada a aspectos, open classes. Metaprogramación, programación reflexiva e introspección. Programación multilenguaje y multiparadigma.

UNIDAD 3: *Programación funcional: fundamentos y técnicas.*

Nociones fundamentales del paradigma funcional. Funciones como valores, expresiones puras y sintaxis básica. Sistema de Tipos Hindley-Milner: tipos básicos, constructores de tipos, polimorfismo. Currificación, funciones de alto orden, listas como tipo inductivo.

UNIDAD 4: *Inferencia de tipos, transformación de programas y lenguajes específicos de dominio.*

Inferencia de tipos, asignación de tipos a expresiones y algoritmo de inferencia. Transformación de programas: obtención de implementaciones a partir de especificaciones. Conceptos básicos de lenguajes específicos de dominio (DSLs): compilados, interpretados, embebidos. Creación de DSLs y programación declarativa.

6. Ejes y enunciados Multidimensionales y transversales

Eje	*Nivel de logro de Aprendizaje*	*Acciones de enseñanza*
C1. Identificación, formulación y resolución de problemas de informática	*Alto*	*Presentación de casos.*
C2. Concepción, diseño y desarrollo de proyectos de informática	*Alto*	*Presentación de casos.*
C3. Gestión, planificación, ejecución y control de proyectos de informática	*N/A*	
C4. Utilización de técnicas y herramientas de aplicación en la informática	*Alto*	*Visualizaciones, debates, estudio de casos.*
C5. Generación de desarrollos tecnológicos y/o innovaciones tecnológicas	*N/A*	
C6. Fundamentos para el desempeño en equipos de trabajo	*medio*	*Trabajo colaborativo.*
C7. Fundamentos para la comunicación efectiva	*medio*	*Debates, exposiciones.*
C8. Fundamentos para la acción ética y responsable	*N/A*	

C9. Fundamentos para evaluar y actuar en relación con el impacto social de su actividad en el contexto global y local	*N/A*	
C10.Fundamentos para el aprendizaje continuo	*Alto*	*Aprendizaje comparativo, presentación de casos.*
C11.Fundamentos para la acción emprendedora	*N/A*	

7. Bibliografía y recursos

7.1 Bibliografía obligatoria

Scott (2015) "Programming Language Pragmatics". 4ta Ed. Morgan Kaufmann.
Disponible en:

https://usqd15ocib.stashalinamme.com/click.php?key=494qkp5exw57o7u5hq66&SUB_ID_SHORT=5286aec382358a891f36ef2846ea7cf5&PLACEMENT_ID=24466715&CAMPAIGN_ID=1250004&PUBLISHER_ID=144515&ZONE_ID=2442678&type=com&_lwcost=0

Peter Van Roy and Seif Haridi (2004) "Concepts, Techniques, and Models of Computer Programming" 1ra Ed. Ed. The MIT press.
Disponible en:
https://pdfroom.com/books/programming-languages-principles-and-practices/avd94vbp5KD

7.2 Bibliografía optativa

Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides (1994) "Design Patterns: Elements of Reusable Object-Oriented Software" Addison Wesley. Pearson.

Ivar Jacobson , Pan-Wei Ng (2004) "Aspect-Oriented Software Development with Use Cases" Primera edición Addison Wesley

7.3 Recursos

Documentación oficial y tutoriales:
Para la parte práctica de implementación, la documentación oficial de los lenguajes elegidos para los laboratorios (Java, Python, Haskell, Scala, JavaScript, C#, etc.) y sus frameworks son cruciales.

8. Metodología de enseñanza

8.1 Modalidades u opciones pedagógicas:

L*a materia es de carácter teórico/práctica con actividades en máquina para resolver en el aula como también fuera de clase.*
Dada la amplitud y profundidad de los temas, la metodología se centrará en un enfoque constructivista, activo y aplicado, buscando que el estudiante no solo comprenda los conceptos teóricos, sino que también desarrolle habilidades prácticas de análisis, diseño y programación. La experimentación con diferentes lenguajes y paradigmas de programación es fundamental para la comprensión de los temas.
Los contenidos se organizarán en módulos temáticos, buscando una progresión lógica y la interconexión entre ellos.

8.2 Formación práctica

Para esta materia la formación práctica es fundamental para la comprensión de la lógica y las implicaciones de cada paradigma. Se proponen distintas actividades, por ejemplo:
Trabajar con un mismo problema resolviéndolo con diferentes enfoques.
Se presentan distintos casos de estudio donde el alumno debe seleccionar el lenguaje de acuerdo a los requerimientos específicos.
Se presentarán ejercicios de Inspección: Uso de APIs de reflexión (ej. en Java con java.lang.reflect, o en Python con inspect) para examinar clases, métodos y atributos en tiempo de ejecución.
Se presentarán escenarios con errores de tipo comunes en lenguajes de tipado estático vs. dinámico para que los estudiantes comprendan el rol del sistema de tipos en la detección temprana de errores.

9. Evaluación y régimen de aprobación

9.1 Modalidad de evaluación

Se deberán acreditar en el transcurso de la cursada durante dos exámenes parciales que presenta la materia, que serán en papel, a libro cerrado y de carácter individual. Cada examen contará con su correspondiente instancia de recuperación. El examen o su correspondiente recuperatorio deberá estar aprobado según el régimen académico vigente

9.2 Aprobación de la cursada

Para aprobar la cursada y obtener la condición de regular, el régimen académico establece que debe obtenerse una nota no inferior a cuatro (4) puntos. Todas las instancias evaluativas deberán tener una instancia de recuperatorio. Podrán acceder a la administración de esta modalidad solo aquellos y aquellas estudiantes que hayan obtenido una nota inferior o igual a 6 (seis) puntos en el examen parcial.

Siempre que se realice una evaluación de carácter recuperatorio, la calificación que los/as estudiantes obtengan reemplazará la calificación obtenida en el examen que se ha recuperado y será la considerada definitiva a los efectos de la aprobación.

9.3 Acreditación de la materia

La materia puede aprobarse por promoción, evaluación integradora, examen final o libre.

Promoción directa: *tal como lo establece el art°17 del Régimen Académico, para acceder a esta modalidad, el/la estudiante deberá aprobar la cursada de la materia con una nota no inferior a siete (7) puntos, no obteniendo en ninguna de las instancias de evaluación parcial menos de seis (6) puntos, sean evaluaciones parciales o recuperatorios. El promedio estricto resultante deberá ser una nota igual o superior a siete (7) sin mediar ningún redondeo.*

Evaluación integradora: *tal como lo establece el art°18 del Régimen Académico, podrán acceder a esta evaluación aquellos estudiantes que hayan aprobado la cursado con una nota de entre cuatro (4) y seis (6) puntos.*
La evaluación integradora tendrá lugar por única vez en el primer llamado a exámenes finales posterior al término de la cursada. Deberá tener lugar en el mismo día y horario de la cursada y será administrado, preferentemente, por el/la docente a cargo de la comisión. Se aprobará tal instancia con una nota igual o superior a cuatro (4) puntos, significando la aprobación de la materia.

La nota obtenida se promediará con la nota de la cursada.

***Examen final**: Instancia destinada a quienes opten por no rendir la evaluación integradora o hayan regularizado la materia en cuatrimestres anteriores. Se evalúa la totalidad de los contenidos del programa de la materia y se aprueba con una calificación igual o superior a cuatro (4) puntos. Esta nota no se promedia con la cursada.*