



Ministerio de Cultura y Educación
Universidad Nacional de San Luis
Facultad de Ciencias Físico Matemáticas y Naturales
Departamento: Informatica
Area: Area IV: Pr. y Met. de Des. del Soft.

(Programa del año 2026)
(Programa en trámite de aprobación)
(Presentado el 07/09/2026 18:55:21)

I - Oferta Académica

Materia	Carrera	Plan	Año	Período
PROGRAMACION III	ING. INFORM.	026/1 2- 08/15	2026	2° cuatrimestre

II - Equipo Docente

Docente	Función	Cargo	Dedicación
SANCHEZ, ALEJANDRO	Prof. Responsable	P.Adj Exc	40 Hs
ZALDUA, ANALIA MAGDALENA	Responsable de Práctico	JTP Simp	10 Hs
ACEVEDO, FACUNDO	Auxiliar de Práctico	A.2da Simp	10 Hs
BAGLIONI, VALENTINA	Auxiliar de Práctico	A.1ra Simp	10 Hs
RIERA BAUER, FABRIZIO JOSÉ	Auxiliar de Práctico	A.2da Simp	10 Hs

III - Características del Curso

Credito Horario Semanal				
Teórico/Práctico	Teóricas	Prácticas de Aula	Práct. de lab/ camp/ Resid/ PIP, etc.	Total
Hs	2 Hs	2 Hs	2 Hs	6 Hs

Tipificación	Periodo
B - Teoria con prácticas de aula y laboratorio	2° Cuatrimestre

Duración			
Desde	Hasta	Cantidad de Semanas	Cantidad de Horas
03/08/2026	13/11/2026	15	90

IV - Fundamentación

Si algo caracteriza al panorama actual del desarrollo de sistemas es su complejidad y heterogeneidad: no existe una sola forma de pensar y encarar las soluciones, no son uniformes los conceptos que fundamentan los lenguajes, no es única la manera de programar. Los modelos de datos, estructuras de control, mecanismos de evaluación, sentencias, enlaces, expresiones, declaraciones y tantos otros elementos que conforman los lenguajes de programación actuales son muy diferentes entre unos y otros, hasta opuestos. Sin embargo, es posible detectar cuáles son los conceptos que marcan diferencias mayores o menores, muestran puntos de contacto o de inflexión, establecen criterios de clasificación, y finalmente, aportan elementos teóricos y prácticos para optimizar el desarrollo de sistemas concretos.

Desde la ingeniería en Sistemas de Información, los lenguajes pueden ser considerados como herramientas para formular, representar y resolver problemas. Desde esta perspectiva, el paradigma que sustenta un lenguaje y la programación en el mismo dicta una forma de modelar o ver un problema. El desarrollo de código en diferentes paradigmas, además de revelar las características de los mismos, permite analizar distintos enfoques con que se puede representar un problema en términos de programación. Esta actividad de desarrollo y análisis conduce a la elección del paradigma más adecuado para resolver

cada problema o por lo menos, si no existe la posibilidad de elección, a una mejor utilización de los recursos del paradigma que se debe utilizar. Existen también los denominados lenguajes multiparadigma. Estos integran y/o soportan características de más de un modelo de desarrollo y permiten su aprovechamiento en el desarrollo de sistemas.

Finalmente, otro componente importante a la hora de programar es el entorno de programación. Nadie discute la relevancia de la Web en el mundo actual. En este contexto, la programación web hace referencia a un ambiente de desarrollo y/o ejecución de programas o servicios en el marco de la Web en general. Existen herramientas, programas, lenguajes de programación y desarrollo que son específicos para el diseño de aplicaciones en este entorno distribuido que son sujetos de estudio del Ingeniero en Informática.

V - Objetivos / Resultados de Aprendizaje

El alumno que complete el curso podrá:

- Utilizar correctamente las singularidades del Paradigma de Programación Lógica y desarrollar aplicaciones con dicho enfoque.
- Aprovechar las posibilidades del Paradigma de Programación Funcional y desarrollar programas en este contexto.
- Reconocer las oportunidades de utilización de la Programación Multiparadigma utilizando correctamente los recursos asociados a cada paradigma.
- Desarrollar aplicaciones web utilizando herramientas y lenguajes de desarrollo específicos para para dicho entorno.

Estos objetivos se conjugan con los los siguientes ejes transversales:

- Identificación, formulación y resolución de problemas de informática.
- Utilización de técnicas y herramientas de aplicación en la informática.
- Fundamentos para el desempeño en equipos de trabajo.
- Fundamentos para la comunicación efectiva.
- Fundamentos para la acción ética y responsable.
- Fundamentos para el aprendizaje continuo.

VI - Contenidos

Contenidos mínimos

- Paradigma de Programación Funcional: El paradigma declarativo y el funcional. Introducción a la programación funcional. Expresiones, tipos, clases de tipos, polimorfismo. Programación con orden superior. Evaluación perezosa e impaciente. Semántica operacional y denotacional. Mónadas.
- Paradigma de Programación Lógica: Introducción. Variables, constantes y estructuras. Átomos, literales y hechos. Reglas. Cláusulas y predicados. Programas, Preguntas y Ejecución. Semántica operacional y árboles lógicos. Unificación. Ejemplos de notación estándar en Prolog. Uso del interfaz de un lenguaje lógico. Aritmética. Factorial. Estructuras internas: functor y arg. Listas explotando Prolog. Uso del 'cut'. Meta predicados. Gramáticas. Lenguaje natural. Contexto del lenguaje.
- Programación Multiparadigma: Características y propiedades. Biblioteca estándar de lenguajes multiparadigma. Tipos de datos, Cadenas de texto. Tuplas y listas. Conjuntos. Controles de flujo. Funciones. Clases. Módulos. Archivos. Lenguajes multiparadigma en la Web. Lenguajes multiparadigmas y bases de datos.
- Programación en entorno Web: Programación en la Web con PHP y MySQL. PHP y Programación Orientada a Objetos. AJAX: Asynchronous JavaScript and XML. PHP y Web Services.

Los contenidos mínimos se desglosan en las unidades que se detallan a continuación.

Unidad I: Paradigma Funcional

Paradigmas declarativos. Semántica operacional y denotacional. Conceptos Generales del paradigma: Características de los lenguajes funcionales. Evaluación perezosa vs Evaluación impaciente. Transparencia referencial. Concepto de función, definición y aplicación. Coincidencia de patrones (Pattern matching). Funciones recursivas. Funciones de orden superior, definición y aplicación. Functor. Mónada. Currificación y aplicación parcial de funciones. Estilo de programación libre de

puntos (pointfree) vs atada a puntos (pointwise). Composición de funciones. Manejo de listas. Listas por comprensión. Lenguaje a utilizar: Haskell. Conceptos particulares del paradigma en el lenguaje Haskell: Entorno de trabajo, definición de programas, uso del intérprete. Módulos. Notación de listas por comprensión. Operadores infijos y prefijos. Reglas de precedencia. Funciones predefinidas para manejo de listas y tuplas. Definición y uso de Funciones de orden superior. Expresiones Lambda. Definición y manipulación de funciones y tipos polimorficos. Análisis de funciones recursivas. Recursión primitiva y estructural. Recursión de cola vs recursión controlada, uso de acumuladores. Análisis y uso de las funciones de folding predefinidas para listas: foldl, foldl1, foldr, foldr1.

Unidad II: Paradigma Lógico

Introducción. Variables, constantes y estructuras: Desarrollo de ejemplos. Átomos, literales y hechos. Reglas. Cláusulas de Horn. Predicados. Programas, Preguntas y Ejecución: Desarrollo de ejemplos. Semántica operacional y árboles lógicos. Unificación: Concepto, ejemplos y ejercicios. Unificador más general. Resolución SLD. Árbol de búsqueda. Ejemplos de notación estándar en Prolog. Uso del interfaz de Prolog: Desarrollo mediante ejemplos en laboratorio. Aritmética. Factorial. Estructuras internas: functor y arg. Modos de funcionamiento de las reglas y los argumentos. Ejemplos de su utilización. Recursión de cola y a izquierda. Listas explotando Prolog. Uso del 'cut'. Corte verde y rojo. Falla. Negación. Metapredicados.

Unidad III: Programación Multiparadigma

Concepto de Programación Multiparadigma: Ejemplos. Características y propiedades. Lenguajes multiparadigma en la Web. Lenguajes multiparadigmas y bases de datos. Biblioteca estándar de lenguajes multiparadigma. Java y Java Streams como ejemplo.

Unidad IV: Programación Web

Internet y los fundamentos de la Web. Arquitecturas de software Web. Arquitectura cliente-servidor. Identificación de recursos web: URI, URL y URN. El protocolo de comunicación web: HTTP. Representación de recursos: el lenguaje HTML. El DOM (Document Object Model). CSS (Cascading Style Sheets). Servidores Web. Virtualización: Docker. Lenguajes de programación de aplicaciones web. Frameworks para la programación de SPA (Single Page Applications). El lenguaje PHP: características generales, sintáxis, variables del sistema, y manejo de formularios.

VII - Plan de Trabajos Prácticos

Metodología de enseñanza

La metodología de enseñanza se fundamenta en los puntos que siguen.

- La teoría se dicta siguiendo una modalidad expositiva-participativa. Se emplean técnicas de trabajo compartido entre estudiantes y docentes, y se propicia la colaboración entre los participantes, tanto como el estudio personal y autónomo.
- Se impulsa a que los estudiantes recurran a la bibliografía de diferentes autores para la profundización, maduración y actualización del conocimiento. Con esta práctica se pretende que los estudiantes perciban las distintos enfoques propuestas por distintos autores para una misma problemática, promoviendo la permanente capacitación sobre temas específicos de la materia.
- Se desarrollan trabajos prácticos individuales y trabajos de laboratorio grupales con la finalidad de profundizar los temas vistos en la materia aplicando los recursos de los distintos paradigmas. Cada práctico incluye problemas básicos para el paradigma así como ejercicios avanzados.
- Se realiza evaluación formativa solicitando la resolución de ejercicios a través de la herramienta Moodle – Aula Virtual, la cual permite brindar feedback de manera inmediata y automática. Esta forma de evaluación posibilita que el estudiante establezca una medida de apropiación del conocimiento. Los datos obtenidos de la herramienta se utilizan para detectar dificultades en el proceso.
- Se procura que el estudiante adopte la conceptualización de cada paradigma y se exprese utilizando un vocabulario acorde, discuta con sus pares sus respuestas y realice autocrítica de su nivel de comprensión.

Organización de la práctica

1. Programación funcional en lenguaje Haskell
2. Programación lógica en lenguaje Prolog
3. Programación multiparadigma en Java con Java Streams
4. Programación Web
5. Laboratorio

Ejes transversales

Identificación, formulación y resolución de problemas de informática

¿Cómo se aborda?

Durante la teoría se resuelven problemas de forma participativa que ilustran las ventajas y desventajas de cada paradigma de programación. Los trabajos prácticos retoman estos problemas desde niveles básicos hasta avanzados. El laboratorio solicita resolver un problema de nivel intermedio que integra conceptos de un paradigma y las tecnologías asociadas.

¿Cómo se evalúa?

Se realiza evaluación formativa y sumativa. La evaluación del tipo formativo se materializa principalmente a través de ejercicios prácticos presentados a en una herramienta que evalúa la respuesta e informa a cada alumno su calificación. Durante las teorías también se solicita la resolución de problemas de manera participativa posibilitando que estudiantes apliquen conceptos de manera inmediata y brindando feedback sobre las distintas soluciones propuestas por ellos. Ambas situaciones permiten a los estudiantes conocer su nivel de apropiación de los contenidos conforme se avanza sobre cada uno de ellos y proporciona a los docentes feedback sobre temas y estudiantes específicos sobre los cuales se debe enfocar el esfuerzo. La evaluación sumativa se realiza a través de parciales, sus respectivas recuperaciones, y el trabajo de laboratorio.

Utilización de técnicas y herramientas de aplicación en la informática

¿Cómo se aborda?

Para cada paradigma y lenguaje presentado se presentan las técnicas y herramientas a utilizar haciendo demostraciones de las mismas en la teoría y reforzando su puesta en acción durante las clases prácticas. Estas técnicas y herramientas permiten verificar las soluciones propuestas a los problemas planteados por la cátedra.

¿Cómo se evalúa?

Se comprueba durante las clases prácticas que cada alumno esté utilizando las herramientas correspondientes.

Fundamentos para el desempeño en equipos de trabajo

¿Cómo se aborda?

El laboratorio se realiza en grupos de hasta tres alumnos.

¿Cómo se evalúa?

Durante el desarrollo del laboratorio se verifica que cada integrante del grupo esté al tanto de cada artefacto desarrollado. Se solicita a cada alumno defender de manera individual lo presentado indicando como se organizó el trabajo y cual fue su participación.

Fundamentos para la comunicación efectiva

¿Cómo se aborda?

Durante las clases los estudiantes deben participar de manera oral en la resolución de ejercicios y socializar sus soluciones

utilizando los conceptos introducidos al avanzar sobre cada paradigma.

¿Cómo se evalúa?

Se observa y corrige durante la participación de cada alumno en la resolución de problemas.

Fundamentos para la acción ética y responsable

¿Cómo se aborda?

Al inicio del cuatrimestre se comparte la agenda de actividades. De esta forma los estudiantes pueden organizar sus horarios y planificar su cursada.

¿Cómo se evalúa?

Los ejercicios prácticos y laboratorio deben entregarse en tiempo y forma. Se verifica que cada alumno haya producido el material entregado y en el caso del laboratorio que todos los integrantes hayan contribuido proporcionalmente.

Fundamentos para el aprendizaje continuo

¿Cómo se aborda?

Se proporciona material para cada paradigma presentado el cual se complementa con fuentes de conocimiento complementaria. Se hace hincapié en que como profesionales en disciplinas de rápida evolución es fundamental desarrollar la capacidad de aprender continuamente y adaptarse.

¿Cómo se evalúa?

Se plantean problemas enfocados en las tecnologías y herramientas a resolver en plazos determinados. Se verifica que cada alumno haya resuelto el problema en el plazo y se realiza una explicación general una vez cumplido el tiempo límite.

VIII - Regimen de Aprobación

Condiciones para regularizar la materia:

- Aprobar los parciales o sus recuperaciones. Se tomarán dos recuperaciones de acuerdo a la OCS 32/14.
- Presentar en tiempo y forma y aprobar el/los laboratorio/s.
- Presentar en tiempo y forma las evaluaciones On-Line solicitadas por la cátedra en el Aula Virtual de la materia.
- Participar en tiempo y forma en los foros de discusión abiertos por la cátedra en el aula Virtual de la materia.
- Asistir al 60% de las clases correspondientes al crédito horario.

Condiciones para promocionar la asignatura:

- Cumplir con las condiciones de regularización con un nivel superior o igual al 70% del total.
- Aprobar la sección especial de integración de cada parcial o su recuperación.
- Asistir al 80% de las clases correspondientes al crédito horario.

El EXAMEN FINAL REGULAR se tomará como un examen escrito u oral en las mesas habilitadas del calendario académico.

El EXAMEN FINAL LIBRE se tomará según lo dispuesto por Art. 27 de Ord. 13/03 CS.

IX - Bibliografía Básica

[1] Anton Antich. Magical Haskell: A Friendly Approach to Modern Functional Programming, Type Theory, and Artificial

Intelligence. Apress, 2025.

[2] Ivan Bratko. Prolog Programming for Artificial Intelligence. Addison-Wesley, 4th edition, 2024.

[3] William F. Clocksin and Christopher S. Mellish. Programming in Prolog. Springer-Verlag, 1981.

[4] Maurizio Gabbriellini and Simone Martini. Programming Languages: Principles and Paradigms. Undergraduate Topics in Computer Science. Springer, Cham, 2 edition, 2023.

[5] Graham Hutton. Programming in Haskell. Cambridge University Press, 2016.

[6] V. Subramaniam. Functional Programming in Java: Harnessing the Power of Java 8 Lambda Expressions. Pragmatic programmers. Pragmatic Bookshelf, 2014.

[7] Peter Van Roy and Seif Haridi. Concepts, techniques, and models of computer programming. MIT press, 2004.

X - Bibliografía Complementaria

[1] Richard Bird. Thinking Functionally with Haskell. Cambridge University Press, 2015.

[2] Richard Bird and Jeremy Gibbons. Algorithm Design with Haskell. Cambridge University Press, 2020.

[3] Philipp Hagenlocher. Learn Haskell by Example. Manning Publications, 2025.

[4] R. Johnson. Prolog Programming Foundations: Definitive Reference for Developers and Engineers. HiTeX Press, 2025.

[5] Kent D. Lee. Foundations of Programming Languages. Undergraduate Topics in Computer Science. Springer, Cham, 2017.

[6] Rebecca Skinner. Haskell Brain Teasers. Pragmatic Bookshelf, 2026.

[7] Simon Thompson. Haskell: The Craft of Functional Programming. Ics Series Addison Wesley, 2011.

[8] Álvaro Vilanova and Eliecer García Rey. ¡aprende haskell por el bien todos! Adaptación de Learn Haskell for the Greater Good! <http://aprendehaskell.es> (visitado 2026)

XI - Resumen de Objetivos

Al finalizar el curso se espera que el estudiante sea capaz de:

- Desarrollar aplicaciones en el Paradigma de Programación Lógica.
- Aprovechar el Paradigma de Programación Funcional y construir programas en este contexto.
- Reconocer las oportunidades de utilización de la Programación Multiparadigma utilizando correctamente sus ventajas.
- Desarrollar aplicaciones Web distinguiendo y aprovechando correctamente los conceptos y herramientas asociados.

XII - Resumen del Programa

Paradigma de Programación Funcional.

Paradigma de Programación Lógica.

Programación en entorno Web.

Programación Multiparadigma.

XIII - Imprevistos

Mail Prof. Responsable: aljsanchez de Gmail

XIV - Otros

Por cualquier consulta ponerse en contacto con el profesor responsable a la cuenta aljsanchez de Gmail.

ELEVACIÓN y APROBACIÓN DE ESTE PROGRAMA	
	Profesor Responsable
Firma:	
Aclaración:	
Fecha:	