IDENTIFICACIÓN DE LA ASIGNATURA

NOMBRE DE ASIGNATURA	DESARROLLO FORMAL DE SOFTWARE		**CÓDIGO**	IS0405	
ÁREA DE FORMACIÓN	FORMACION ESPECIFICA		**MODALIDAD**	PRESENCIAL	
CRÉDITOS	3		**HABILITABLE**	N/A	
PROGRAMA (S)	INGENIERIA DE SISTEMAS		**VALIDABLE**	SI	
SEMESTRE	06		**PRERREQUISITOS**	INGENIERÍA DE SOFTWARE	
INTENSIDAD HORARIA (Horas Semanales)	PRESENCIAL	4	TRABAJO INDEPENDIENTE		5
UNIDAD ACADÉMICA	PROGRAMA DE INGENIERIA DE SISTEMAS				

DESCRIPCIÓN DE LA ASIGNATURA

ASPECTOS ACADÉMICOS DE LA ASIGNATURA	
PRESENTACIÓN	Desarrollo Formal de Software es una asignatura orientada a la construcción de software mediante prácticas de programación, especificación, verificación, pruebas y evaluación de la calidad. La asignatura articula los conocimientos adquiridos en Análisis de Sistemas, Diseño de Sistemas e Ingeniería de Software. A partir de una funcionalidad seleccionada, el estudiante construye software ejecutable, establece condiciones y propiedades que debe cumplir, incorpora mecanismos de especificación y contratos, aplica técnicas de verificación y pruebas, identifica defectos y propone mejoras de calidad. El curso busca que el estudiante evolucione de la comprobación basada exclusivamente en ejemplos hacia una práctica de desarrollo en la que pueda especificar qué debe cumplir un programa, construirlo y obtener evidencias de su correctitud y calidad.
JUSTIFICACIÓN	La ingeniería de software requiere desarrollar sistemas que no solamente respondan a los requerimientos establecidos, sino que presenten comportamientos previsibles, verificables y confiables. En este contexto, los métodos y técnicas formales permiten expresar de manera precisa propiedades y condiciones que debe cumplir un programa, proporcionando mecanismos para razonar sobre su comportamiento y detectar inconsistencias o defectos. De esta manera, el estudiante desarrolla la capacidad de pasar de un requerimiento y una especificación a una implementación que pueda ser analizada y verificada, fortaleciendo la confiabilidad, mantenibilidad y calidad del software.

COMPETENCIAS QUE DESARROLLA LA ASIGNATURA		
PERSONALES	SER	Ciudadanos éticos, responsables, comprometidos con su comunidad, con el desarrollo sostenible y sustentable, con el reconocimiento de su identidad cultural y de los valores humanos

	SABER	• Desempeñarse de manera efectiva en equipos de trabajo. • Identificar, formular y resolver problemas de ingeniería. • Concebir, diseñar y desarrollar proyectos de ingeniería. • Utilizar de manera efectiva las técnicas y herramientas de aplicación en la ingeniería. • Aprender en forma continua y autónoma.
	SABER HACER	• Planificar estrategias para el análisis y la resolución de situaciones problema, a partir de la identificación de los datos, la representación de los mismos y el establecimiento de relaciones, integrando los saberes de las ciencias, las matemáticas y las ciencias básicas de la ingeniería. • Implementar metodologías adecuadas para la gestión, planificación y desarrollo de proyectos de software.
GENERALES	Desarrollar software mediante técnicas de especificación, programación, verificación y pruebas que permitan establecer y comprobar el cumplimiento de propiedades y condiciones de correctitud y calidad.	
ESPECIFICAS	Comprender los fundamentos del desarrollo formal, la correctitud, la verificación, la validación, las pruebas y la calidad del software.	
	Formalizar propiedades y condiciones de comportamiento mediante precondiciones, postcondiciones, aserciones, invariantes y contratos.	
	Construir programas y componentes de software a partir de especificaciones previamente establecidas.	
	Aplicar técnicas de verificación de programas, incluyendo lógica de Hoare, invariantes, razonamiento sobre correctitud y terminación.	
	Utilizar herramientas de verificación automatizada para analizar propiedades de programas y reconocer contraejemplos o condiciones no satisfechas.	
	Aplicar pruebas unitarias, pruebas de casos límite y técnicas complementarias de análisis de calidad.	
	Integrar especificación, construcción, verificación, pruebas y calidad en una funcionalidad del proyecto de software.	
ALCANCES ESPERADOS	Al finalizar la asignatura, el estudiante estará en capacidad de construir una funcionalidad de software a partir de especificaciones previamente establecidas, expresar formalmente condiciones de comportamiento mediante contratos, precondiciones, postcondiciones e invariantes, aplicar técnicas de verificación y pruebas, utilizar herramientas de apoyo a la verificación y analizar la calidad del software desarrollado, documentando y sustentando las evidencias obtenidas.	

CONTENIDOS MÍNIMOS

UNIDAD	CAPITULO	TEMAS
1. FUNDAMENTOS DEL DESARROLLO FORMAL Y CALIDAD DEL SOFTWARE	**1. INTRODUCCIÓN AL DESARROLLO FORMAL**	Concepto de desarrollo formal. Desarrollo de software y desarrollo formal. Correctitud del software. Error, defecto y fallo. Verificación y validación. Pruebas de software. Relación entre especificación, construcción, verificación y pruebas.
	2. CALIDAD Y CONSTRUCCIÓN DE SOFTWARE	Calidad del producto software. Calidad funcional y técnica. Confiabilidad. Mantenibilidad y legibilidad. Robustez. Revisión del código. Casos límite y condiciones de error. Buenas prácticas para la construcción de software.
2. ESPECIFICACIÓN FORMAL Y DESARROLLO BASADO EN CONTRATOS	**3. FUNDAMENTOS DE ESPECIFICACIÓN FORMAL**	Concepto de especificación formal. Propiedades y condiciones de un programa. **Lógica proporcional** Proposiciones + operadores + Tablas de verdad **Lógica de predicados** Predicados + Variables + Cuantificadores **Formalización** Expresiones booleanas. Formalización de condiciones.
	4. ESPECIFICACIÓN BASADOS EN CONTRATOS	**Condiciones de una operación** Precondiciones. Postcondiciones. Entradas y salidas. **Propiedades durante la ejecución** Aserciones. Invariantes. Condición de error. **Contratos** Contratos de funciones/métodos. Contratos de clases. Diseño por contrato. Refinamiento de especificaciones.

3. VERIFICACIÓN FORMAL DE PROGRAMAS	**5. FUNDAMENTOS DE VERIFICACIÓN DE PROGRAMAS**	Concepto de verificación formal. Verificación vs pruebas. Correctitud parcial. Correctitud total. Estados de un programa. Aserciones. Precondiciones y postcondiciones. Tripletas de Hoare. Lógica de Hoare. Razón de correctitud.
	6. VERIFICACIÓN DE ESTRUCTURAS DE PROGRAMAS	Verificación de asignaciones. Verificación de secuencias. Verificación de estructuras condicionales. Verificación de ciclos. Invariantes de ciclos. Terminación de algoritmos. Verificación de funciones. Verificación de algoritmos. Recursividad. Contraejemplos. Corrección de errores.
	7. VERIFICACIÓN DE PROGRAMAS ORIENTADOS A OBJETOS	Especificación de clases. Contratos de métodos. Invariantes de clase. Relaciones entre objetos. Herencia y contratos. Verificación modular. Verificación de componentes de software.

4. PRUEBAS, VERIFICACIÓN AUTOMATIZADA Y CALIDAD	**8. PRUEBAS DE SOFTWARE**	Pruebas unitarias. Pruebas funcionales. Pruebas de casos límite. Pruebas negativas. Pruebas basadas en propiedades. Diseño de casos de prueba. Pruebas de robustez. Relación entre pruebas y verificación formal.
	9. VERIFICACIÓN AUTOMATIZADA	Automatización de la verificación. Verificación estática. Herramientas de verificación. Solucionadores SAT y SMT. Contraejemplos. Verificación basada en contratos. Introducción al model checking. Propiedades de seguridad. Propiedades de vivacidad. Introducción a lógica temporal.
	10. PROYECTO DE DESARROLLO Y VERIFICACIÓN	Selección de una funcionalidad del proyecto. Especificación. Construcción del programa. Incorporación de contratos. Verificación. Pruebas. Identificación y corrección de defectos. Evaluación de calidad. Documentación de evidencias. Socialización del proyecto.

SALIDA DE CAMPO O PEDAGÓGICA	**SI**		**NO**	**X**

EVALUACIÓN

La evaluación es un proceso continuo cuyo objetivo principal es valorar las habilidades y destrezas adquiridas por los estudiantes.

A lo largo del curso, se aplicarán diferentes instrumentos de evaluación con el fin de obtener una calificación cuantitativa que de alguna manera interprete el trabajo desarrollado por cada Estudiante; sin embargo, dicha calificación no será el criterio final de aprobación del curso ya que se tendrán en cuenta aspectos como: Asistencia, participación en clases, responsabilidad, puntualidad, responsabilidad, respeto, etc.

Las actividades evaluativas están divididas en 3 cortes:

1er corte = 30 %

2do corte = 30 %

3er corte = 40 %

La nota en cada uno de los cortes se obtiene a partir de un conjunto de actividades evaluativas: examen parcial, actividades complementarias (quiz, taller, exposición, etc.), proyecto de curso. En el acuerdo pedagógico se define el porcentaje de cada una de estas actividades dentro de cada corte.

BIBLIOGRAFÍA

DISPONIBLE EN BIBLIOTECA				
LIBROS				
AUTOR(ES)	TITULO	EDICIÓN	AÑO	EDITORIAL
OTROS (PÁGINAS WEB, ARTÍCULOS, REVISTAS, MEDIOS ÓPTICOS ETC.)				

NO DISPONIBLE EN LA BIBLIOTECA				
LIBROS				
AUTOR(ES)	TITULO	EDICIÓN	AÑO	EDITORIAL
Leino, K. Rustan M.	Program Proofs.	The MIT Press,	2023.	
Pressman, Roger S.; Maxim, Bruce R.	Ingeniería de software: un enfoque práctico.	9.ª edición.		McGraw-Hill.
OTROS (PÁGINAS WEB, ARTÍCULOS, REVISTAS, MEDIOS ÓPTICOS ETC.)				

ESTADO LEGAL INTERNO Y CONTROL DE SEGUIMIENTO/CAMBIOS DE LA ASIGNATURA

ELABORACIÓN			
ELABORARON	Yowanna Karina Caicedo Guerrero	REVISARON	
FECHA	Agosto 2026	FECHA	

REVISIONES/CAMBIOS		
	AUTOR	FECHA