Programa de Asignatura

1. Identificación Asignatura

Nombre:	Ingeniería de Software			**Código:**	
Carrera:	Ingeniería Civil Informática	**Unidad Académica:**		Ciencias Naturales y Tecnología	
Ciclo Formativo:	Ciclo Licenciatura	**Línea formativa:**		Especializada	
Semestre	VI	**Tipo de actividad:**		Obligatoria	
N° SCT:	6	**Horas Cronológicas Semanales**			
		Presenciales:	4.5	**Trabajo Autónomo:**	4.5
Pre-requisitos	Bases de Datos				

2. Propósito formativo

El curso Ingeniería de Software tiene como propósito que los y las estudiantes adquieran los conceptos fundamentales sobre diferentes enfoques de desarrollo de software, dependiendo de los requisitos y la complejidad de la solución, reconociendo las etapas involucradas en el desarrollo de software.

En su primera parte, la asignatura incorpora elementos conceptuales de la ingeniería de software, reconociendo las distintas etapas del ciclo de vida de desarrollo. Esto además a través de diferentes modelos, como cascada, espiral, iterativo, entre otros. Luego, los y las estudiantes aprenderán elementos y técnicas sobre captura de requerimientos, diseño, implementación, verificación y validación de software.

Esta asignatura es un curso clásico y fundamental, y será prerrequisito para los cursos Gestión de Proyectos Informática, y Taller de Desarrollo Avanzado de Software del semestre VII.

3. Contribución al perfil de egreso

Esta asignatura contribuye a los siguientes desempeños o resultados de aprendizaje globales declarados en el Perfil de Egreso de la carrera:

1. Desarrolla productos y servicios de software, a través de metodologías, de procesos analíticos y de diseño, que consideren las características de las distintas plataformas y lenguajes disponibles, para abordar necesidades de diversos usuarios.
2. Gestiona proyectos de software, utilizando diversos métodos, para que el proceso de desarrollo y la operación de los sistemas se realice de manera eficiente y eficaz.
3. Demuestra compromiso con la realidad social, cultural y medioambiental de la región de Aysén.

4. Resultados de aprendizaje específicos

Resultado de Aprendizaje Específico	Criterios de evaluación	Evidencia
RA1. Conoce etapas y características del ciclo de vida del desarrollo de software a través de diferentes modelos (incremental, iterativo, cascada, etc)	1.1. Maneja conceptos de software, ingeniería, ciencias de la computación e ingeniería de software. 1.2. Reconoce principios generales para el desarrollo de software de calidad. 1.3. Identifica las etapas en el desarrollo de software. 1.4. Conoce diferentes modelos de desarrollo de software.	Guías de ejercicio, informes,
RA2. Aprende a identificar adecuadamente los requerimientos del software y diseña soluciones a partir de su definición.	2.1. Aprende conceptos de requerimientos del software. 2.2. Utiliza técnicas para especificar: funcionalidad, requerimientos, operaciones y datos. 2.3. Aplica distintas estrategias de diseño para distintos módulos/secciones del software.	Guías de ejercicio, Taller, Evaluaciones teóricas
RA3. Verifica y valida el software a través de pruebas funcionales de una solución a una problemática vinculada a la realidad social, cultural y medioambiental de la región de Aysén.	3.1. Utiliza diferentes modelos de verificación del software. 3.2. Realiza pruebas de caja negra, pruebas de caja blanca, pruebas de integración. 3.3. Aplica verificación estática.	Laboratorios, guías de ejercicio.

5. Unidades de Aprendizaje

1. Introducción a la Ingeniería de Software y Metodologías de Desarrollo 1.1. Conceptos generales de ingeniería de software 1.2. Modelos de ciclo de vida del desarrollo de software. 1.3. Modelo de cascada, iterativo, incremental, espiral, XP.
2. Requerimientos y Diseño de Software 2.1. Concepto de requerimientos del software. 2.2. Técnicas para especificación de requerimientos. 2.3. Diseño arquitectónico del software. 2.4. Patrones de arquitectura 2.5. Usabilidad / experiencia de usuario y diseño de interfaces con el usuario. 2.6. Diseño orientado a objetos.

3. Verificación y Validación del Software

3.1. Conceptos de verificación y validación de software.
3.2. Pruebas funcionales, estructurales y de integración.
3.3. Verificación estática del software.

6. Recursos de Aprendizaje

Bibliografía:
B1: Fundamentals of Software Engineering (2nd Edition), Carlo Ghezzi, Mehdi Jazayeri, Dino Mandrioli, Prentice Hall; September 29, 2002.
https://www.abebooks.com/9787508338767/Fundamentals-Software-Engineering-2nd-Edition-7508338766/plp

B2: Software Engineering: (Update) (8th Edition), Ian Sommerville; Addison Wesley, June 4, 2006
https://doc.lagout.org/science/0_Computer%20Science/Software%20Engineering%2C%208th%20Edition.pdf

B3 bibliografía p.e. PRESSMAN (RA2)

Recursos materiales e infraestructura: ●
Laboratorio de computación.
- Acceso a Ucampus.
- Acceso a Googlesites con credenciales institucionales.

Computadores debidamente equipados para utilizar lenguajes de alto nivel (por ej.: Python).

7. Comportamiento y ética académica:

Se espera que los estudiantes actúen en sus diversas actividades académicas y estudiantiles en concordancia con los principios de comportamiento ético y honestidad académica propios de todo espacio universitario y que están estipulados en el *Reglamento de Estudiantes de la Universidad de Aysén*, especialmente aquéllos dispuestos en los artículos 23°, 24° y 26°.
Todo acto contrario a la honestidad académica realizado durante el desarrollo, presentación o entrega de una actividad académica del curso sujeta a evaluación, será sancionado con la suspensión inmediata de la actividad y con la aplicación de la nota mínima (1.0).

PLANIFICACION DE ASIGNATURA

1. Responsables

Académico (s) Responsable (s) y equipo docente	**Profesor**: Mg. Claudio Marcos Levicán Jaque		
Contacto	**Correo**: clevicanj@hotmail.com		
Año	2024	**Periodo Académico**	Segundo Semestre
Horario clases	Cátedra: Lunes 18:00 a 19:30 Martes 18:00 a 19:30 Jueves 18:00 a 19:30	**Horario de atención estudiantes**	Horario por confirmar con los/las estudiantes
Sala / Campus	Sala Ucampus		

2. Metodología de Trabajo:

La asignatura contiene:			
Actividades de vinculación con el medio	**No**	Actividades relacionadas con proyectos de investigación	**No**

En el curso se contemplan cuatro tipos de actividades docentes, las cuales se asocian a requerimientos de sala y al nivel de intervención del profesor:

Actividad docente	Descripción	Intervención del profesor/ayudante	Requerimiento de sala
Exposición conceptual	El profesor introduce conceptos de programación preliminares y necesarios a otras actividades de índole práctica, de forma expositiva. Se dispone de materiales complementarios en la plataforma Ucampus.	Alta	Sala de clases
Programación expositiva	El profesor profundiza en la comprensión de elementos conceptuales a través de la exposición directa de la resolución de problemas de programación como ejemplos.	Alta	Sala de clases
Programación tutorial	Funciona como la programación expositiva, pero el profesor realiza pausas para que los alumnos completen “pasos requeridos” antes de continuar. El objetivo es que todos los alumnos completen un paso definido por el profesor antes de continuar al siguiente.	Media	Sala de clases
Actividad práctica / Programación autónoma	Los estudiantes abordan y resuelven problemas de programación de forma autónoma, algunas con guía y apoyo docente y otras no.	Baja/Media	Sala de clases

En cualquier semana del semestre en curso se podría realizar una **evaluación menor** sobre las temáticas estudiadas a la fecha. Esta evaluación menor puede ser de los siguientes tipos:

- **Laboratorio**: Evaluación individual o grupal, que se realiza en el computador. Ocupará los bloques del día jueves.
- **Guía de ejercicios**: Evaluación individual que se realiza en computador durante el tiempo de trabajo autónomo.
- ***Prueba Parcial:*** Evaluación individual que se realiza en computador en el horario de clases.
- ***Proyecto:*** Evaluación individual o grupal, que se realiza en el computador. Ocupará los bloques horario autónomo.

3. Evaluaciones:

Evaluación	Ponderaciones específicas	Ponderación nota presentación
Pruebas de cátedra	Prueba 1 : semana 3 10% . Trabajo con Exposición (P2): semana 4, 20%.	30%
Trabajos	Trabajo semana 8 10% Trabajo Semana 10 20%	30%
Proyectos	Informe Proyecto Semana 10 20% Presentación proyecto 20%	40%

Calificación final:
Nota de presentación: 70%
Examen Final: 30 %

Condiciones de eximición:
Nota de presentación igual o superior a nota 5,0
Ponderación específica pruebas de cátedra >= 4,0

Derecho a rendir examen:
Nota de presentación >= 3,5

Requisito de Aprobación
Asistencia: 70%
Nota Final: 4,0

Semana / Sesión	Resultado(s) de Aprendizaje	Tema (Unidades de aprendizaje) y actividades	Recursos utilizados o lecturas	Actividad(es) de Trabajo Autónomo
Semana 1	RA1: Conoce etapas y características del ciclo de vida del desarrollo de software.	Introducción a la Ingeniería de Software	Charla introductoria sobre conceptos básicos de ingeniería de software.	Material de lectura sobre conceptos de ingeniería de software.
Semana 2	RA1: Conoce etapas y características del ciclo de vida del desarrollo de software.	Modelos de ciclo de vida del desarrollo de software (cascada, iterativo)	Presentación y análisis de modelos de ciclo de vida: cascada e iterativo.	Presentaciones de Investigación sobre modelos de ciclo de vida.
Semana 3	RA1: Conoce etapas y características del ciclo de vida del desarrollo de software.	Modelos de ciclo de vida (incremental, espiral, XP)	Profundización en modelos de ciclo de vida: incremental, espiral y XP.	Comparativas sobre diferentes modelos de ciclo de vida.
Semana 4	RA1: Conoce etapas y características del ciclo de vida del desarrollo de software.	Comparativa y selección de modelos de ciclo de vida	Discusión grupal sobre selección de modelos según contexto del proyecto.	Casos de estudio sobre selección de modelos.
Semana 5	RA2: Aprende a identificar adecuadamente los requerimientos del software y diseña soluciones a partir de su definición.	Concepto de requerimientos del software	Introducción a los conceptos de requerimientos del software. Taller de técnicas de elicitación de requerimientos.	Materiales sobre técnicas de elicitación de requerimientos.
Semana 6	RA2: Aprende a identificar adecuadamente los requerimientos del software y diseña soluciones a partir de su definición.	Técnicas de especificación de requerimientos	Taller de elaboración de un documento de requerimientos utilizando casos de uso.	Ejemplos de casos de uso y especificación de requerimientos.
Semana 7	RA2: Aprende a identificar adecuadamente los requerimientos del software y diseña soluciones a partir de su definición.	Diseño arquitectónico del software	Introducción al diseño arquitectónico y patrones de diseño. Taller de diagramas UML: casos de uso y clases.	Guías sobre diagramas UML y patrones de diseño.
Semana 8	RA2: Aprende a identificar adecuadamente los requerimientos del software y diseña soluciones a partir de su definición.	Patrones de diseño y diagramas UML	Continuación del taller de diagramas UML: secuencia. Aplicación de patrones de diseño en un proyecto práctico.	Herramientas de diseño de software
Semana 9	RA2: Aprende a identificar adecuadamente los requerimientos del software y diseña soluciones a partir de su definición.	Usabilidad y experiencia de usuario (UI/UX)	Introducción a los principios de diseño de UI/UX. Taller de diseño de interfaces utilizando herramientas de prototipado.	Software de prototipado
Semana 10	RA2: Aprende a identificar adecuadamente los requerimientos del software y diseña soluciones a partir de su definición.	Diseño de interfaces con el usuario	Desarrollo de un prototipo de una interfaz de usuario para un sistema propuesto.	Guías sobre diseño de interfaces y prototipos.

Semana / Sesión	Resultado(s) de Aprendizaje	Tema (Unidades de aprendizaje) y actividades	Recursos utilizados o lecturas	Actividad(es) de Trabajo Autónomo
Semana 11	RA3: Verifica y valida el software a través de pruebas funcionales.	Conceptos de verificación y validación	Introducción a pruebas unitarias y de integración. Ejercicio práctico de diseño de pruebas funcionales.	Lecturas sobre verificación y validación de software.
Semana 12	RA3: Verifica y valida el software a través de pruebas funcionales.	Pruebas funcionales, estructurales y de integración	Continuación de pruebas: estructurales y de integración. Revisión de estrategias de verificación estática.	Materiales sobre pruebas de software y verificación estática.
Semana 13	RA3: Verifica y valida el software a través de pruebas funcionales.	Herramientas de testing	Laboratorio con uso de herramientas de testing Introducción a la automatización de pruebas.	Software de testing
Semana 14	RA3: Verifica y valida el software a través de pruebas funcionales.	Automatización de pruebas	Continuación del laboratorio de automatización de pruebas.	Documentación de herramientas de automatización de pruebas.
Semana 15	RA1, RA2, RA3: Integración de conocimientos	Desarrollo de un proyecto integrador	Inicio del desarrollo del proyecto integrador. Seguimiento y presentación de avances.	Guías del proyecto integrador.
Semana 16	RA1, RA2, RA3: Integración de conocimientos	Desarrollo de un proyecto integrador	Finalización y presentación del proyecto.	Recursos del proyecto integrador.
Semana 17	RA1, RA2, RA3: Revisión y evaluación final	Revisión de contenidos del curso	Revisión general de todo el contenido del curso. Resolución de dudas y preparación para el examen final.	presentación de proyecto de software
Semana 18	RA1, RA2, RA3: Revisión y evaluación final	Revisión Examen		