Programa de asignatura
por competencias de educación superior

Sección I. Identificación del Curso

Tabla 1. Identificación de la Planificación del Curso.

Actualización:	Marzo 22, 2022				
Carrera:	Ingeniería en Tecnologías de Software		**Asignatura:**	Arquitectura de Software	
Academia:	Informática y Computación Virtual /		**Clave:**	19SICTS0507	
Módulo formativo:	Informática y Computación		**Seriación:**	19SICTS0705 - Programación Web	
Tipo de curso:	Modalidad mixta		**Prerrequisito:**	19SICTS0405 - Ingeniería de Software	
Semestre:	Quinto	**Créditos:** 5.63	**Horas semestre:**	90 horas	
Teoría:	0 horas	**Práctica:** 0 horas	**Trabajo indpt.:**	5 horas	**Total x semana:** 5 horas

Sección II. Objetivos educacionales

Tabla 2. Objetivos educacionales

Objetivos educacionales		Criterios de desempeño	Indicadores
OE1	Solucionará problemas con sólidas bases científicas y fundamentos tecnológicos que le permitirán comprender, analizar, diseñar, organizar, producir, operar y dar soluciones prácticas a problemas relacionados con las áreas de Organización de Sistemas Computacionales e Ingeniería en Software para el sector productivo y social, promoviendo los principios de ética, responsabilidad y trabajo colaborativo.	El egresado implementará las diferentes etapas del ciclo de vida del software contemplando la protección de datos y prevención de desastres, salvaguardando con ética la seguridad de la información.	50 % Egresados trabajarán en cualquier proceso del desarrollo de software o áreas afines a los sistemas computacionales, promoviendo los principios de ética, responsabilidad y trabajo colaborativo.
OE2	Aportará soluciones innovadoras y sustentables en el área de la electrónica en el que establezca el análisis, diseño, implementación, selección de componentes de hardware de uso específico, el software asociado y su conectividad a través de redes de comunicación para el sector productivo y social.	El egresado implementará las diferentes técnicas de análisis y diseño de circuitos electrónicos que den una solución innovadora sustentable a problemas con el hardware.	20% Egresados trabajarán en cualquier proceso de creación y aplicación de hardware o áreas afines en el sector productivo y social.
OE3	Implementará soluciones innovadoras y sustentables con tecnologías de información que sean acordes a las necesidades, a las tecnologías disponibles y emergentes, para lograr un aprovechamiento óptimo de los recursos humanos y financieros en el sector productivo y social.	El egresado implementará las diferentes tecnologías emergentes en equipos multidisciplinarios que den una solución innovadora y sustentable a las necesidades que se presenten en el ámbito productivo y social.	20 % Egresados trabajarán en la aplicación de Tecnologías de la información o áreas afines en el sector productivo o social.

Atributos de egreso de plan de estudios		Criterios de desempeño	Componentes
AE1	Aplicar los principios físicos-matemáticos y de las ciencias de la ingeniería para crear soluciones de software eficientes e innovadoras en los ámbitos industrial y empresarial.	CD1. Comprenderá la arquitectura en capas y orientada a objetos en eldiseño de software utilizando metodologías clásicas y metodologías ágiles que puedan contribuir a la solución de sistemas en el campo de la ingeniería de software.	1. Introducción a la arquitectura y diseño del software. 1.1.1. Metodologías de desarrollo de software. 1.1.2. Arquitectura y diseño de software. 1.1.3. Arquitectura lógica y física de un sistema software. 1.1.4. Rol de los patrones de diseño en el diseño del software. 1.1.5. Arquitectura y diseño del software en metodologías clásicas y ágiles. 2. Arquitectura en capas y orientada a objetos. 2.1.1. Patrón arquitectónico en capas. 2.1.2. Aplicación del patrón arquitectónico en capas a los sistemas y servicios software. 2.1.3. Principios de diseño de la arquitectura en capas. 2.1.4. Patrón arquitectónico orientación a objetos. 2.1.5. Aplicación del patrón arquitectónico orientación a objetos a los sistemas software. 2.1.6. Principios de diseño de la arquitectura orientada a objetos. 3. Diseño del software utilizando metodologías clásicas. 3.1.1. Diseño del software utilizando metodologías clásicas. 3.1.2. Diseño de la capa de dominio. 3.1.3. Patrones Domain Model y Transaction Script. 3.1.4. Patrones de diseño de la capa de dominio.

Continuación: Tabla 2. Objetivos educacionales (continuación			
No.	Atributos de egreso de plan de estudios	Criterios de desempeño	Componentes
			3.1.5. Uso de servicios software. 3.1.6. Diseño de la capa de presentación. 3.1.7. Diseño externo de un sistema software. 3.1.8. Diseño interno de la capa de presentación. 3.1.9. Diseño de la capa de datos. 3.1.10. Tecnología de bases de datos relacionales. 3.1.11. Estrategias de gestión de la persistencia. 3.1.12. Generación automática de la persistencia. 3.1.13. Diseño directo de la persistencia. 4. Diseño de software utilizando metodologías ágiles. 4.1.1. Diseño del software utilizando metodologías ágiles. Extreme Programming. TDD. Tests en TDD. 4.1.2. Coding en TDD. Pair Programming. Simple Design and Incremental design and architecture. Code Smells and Refactorings.

Sección III. Atributos de la asignatura

Tabla 3. Atributos de la asignatura

Problema a resolver		
Dominar y aplicar tipos de arquitecturas en el diseño de software utilizando metodologías clásicas y metodologías ágiles de ingeniería en diferentes contextos.		
Atributos (competencia específica) de la asignatura		
Desarrollar programas en el diseño de software con arquitecturas en capas y orientada a objetos, utilizando metodologías clásicas y ágiles.		
Aportación a la competencia específica		**Aportación a las competencias transversales**
Saber	Saber hacer	Saber Ser
- Conocer la diferentes arquitecturas y metodologías de diseño del software, - Conocer las características y principios de diseño de la arquitectura en capas y orientada a objetos. - Conocer las metodologías clásicas de diseño del software, patrones de diseño, uso de servicios software, tecnología de bases de datos relacionales, estrategias de gestión, generación automática y diseño directo de la persistencia. - Conocer las metodologías ágiles de diseño de software: Extreme Programming. TDD. Test en TDD, Coding en TDD, Pair Programming, Simple Design and Incremental design and architecture, Code Smells and Refactorings.	- Resolver diseño de software utilizando la arquitectura en capas y orientada a objetos, utilizando metodologías clásicas y metodologías ágiles. - Aplicar los conocimientos en la práctica en el desarrollo de diseño de software. -Identificar, plantear y resolver diseño de software específico con arquitectura y diseño del software, en capas y orientada a objetos y diseño del software utilizando metodologías clásicas y ágiles.	- Aporta puntos de vista con apertura a aprender de los otros y considera los de otras personas de manera reflexiva y respetuosa. - Participa activamente en la construcción de su aprendizaje y en la resolución de problemas, colaborando de manera productiva en espacios y equipos de trabajo. - Cumple en tiempo y forma en sus obligaciones como estudiante, siguiendo las indicaciones y considerando los criterios de evaluación. - Utiliza la tecnología para apoyar su aprendizaje y para el desarrollo de habilidades metacognitivas, el aprendizaje autónomo y el aprendizaje learning.
Producto integrador de la asignatura, considerando los avances por unidad		
Proyecto final de diseño de software a partir de metodologías clásicas y ágiles de una entidad de empresa dando solución a la necesidad real de la organización, incorporando las competencias desarrolladas en cada una de las unidades de aprendizaje. Reporte del proyecto final, cumpliendo con los criterios y documentación considerando también los siguientes aspectos: Cumplimiento del objetivo del proyecto de acuerdo a la aplicación. Cumplimiento del tiempo de entrega.		

Sección IV. Desglose específico por cada unidad formativa

Tabla 4.1. Desglose específico de la unidad "Introducción a la arquitectura y diseño del software."

Número y nombre de la unidad:	1. Introducción a la arquitectura y diseño del software.					
Tiempo y porcentaje para esta unidad:	Teoría:	5 horas	Práctica:	12 horas	Porcentaje del programa:	18.89%
Aprendizajes esperados:	Conocer e identificar los componentes y aspectos de la arquitectura y diseño de software, para realizar diseños eficientes que resuelvan problemas de la ingeniería.					

Temas y subtemas (secuencia)	**Criterios de desempeño**	**Estrategias didácticas**	**Estrategias de evaluación**	**Producto Integrador de la unidad (Evidencia de aprendizaje de la unidad)**
1. Introducción a la arquitectura y diseño del software. 1.1.1. Metodologías de desarrollo de software. 1.1.2. Arquitectura y diseño de software. 1.1.3. Arquitectura lógica y física de un sistema software. 1.1.4. Rol de los patrones de diseño en el diseño del software. 1.1.5. Arquitectura y diseño del software en metodologías clásicas y ágiles.	Saber: - Identificar la arquitectura y diseño del software en metodologías de desarrollo de software, arquitectura y diseño de software. - Conocer la arquitectura lógica y física de un sistema de software, rol de los patrones dediseño en el diseño del software y arquitectura y diseño de software en metodologías clásicas y ágiles en el contexto de la ingeniería de software Saber hacer:	- Preguntas intercaladas para evaluar los conocimientos previos. - Exposición por parte del profesor de material teórico. - Complementar información con material audiovisual. - Actividades para estructuración y apropiación de información: análisis de casos, resúmenes y/o mapas conceptuales.	Evaluación diagnóstica: - Rescatar conocimientos previos con preguntas intercaladas. Evaluación Formativa: - Análisis de caso de problemas resueltos. Instrumento de evaluación: - Lista de cotejo y/o rúbrica. Evaluación Sumativa: -Examen escrito. Instrumento de evaluación: - Cuestionario.	Portafolio de evidencias: Actividades realizadas en clase. Análisis de examen escrito.

Continuación: Tabla 4.1. Desglose específico de la unidad "Introducción a la arquitectura y diseño del software."				
Temas y subtemas (secuencia)	Criterios de desempeño	Estrategias didácticas	Estrategias de evaluación	Producto Integrador de la unidad
	- Resolver ejercicios de arquitectura ydiseño del software con metodologías dedesarrollo de software, arquitectura y diseño de software, arquitectura lógica y física de un sistema de software, rol de los patrones de diseño en el diseño del software y arquitectura y diseño de software en metodologías clásicas y ágiles en el contexto de la ingeniería de software. Ser: - Aporta puntos de vista con apertura a aprender de los otros y considera los de otras personas de manera reflexiva y respetuosa.			

Bibliografía

Meszaros, G. (2007). refactoring test code. En XUnit test patterns(pp. 130-145). Boston: Pearson Education.

Pressman, R.S.; Maxim, B.R. (2015). A practitioner's approach. En Software engineering (pp. 140-150). Estados Unidos de América: McGraw Hill.

Sommerville, I. (2016). Software engineering. Estados Unidos de América: Pearson Education.

Sección IV. Desglose específico por cada unidad formativa

Tabla 4.2. Desglose específico de la unidad "Arquitectura en capas y orientada a objetos."

Número y nombre de la unidad:	2. Arquitectura en capas y orientada a objetos.					
Tiempo y porcentaje para esta unidad:	Teoría:	5 horas	Práctica:	10 horas	Porcentaje del programa:	16.67%
Aprendizajes esperados:	- Conocer las características y principios de diseño de la arquitectura en capas y orientada a objetos, para aplicarlas adecuada y eficientemente.					

Temas y subtemas (secuencia)	Criterios de desempeño	Estrategias didácticas	Estrategias de evaluación	Producto Integrador de la unidad (Evidencia de aprendizaje de la unidad)
2. Arquitectura en capas y orientada a objetos. 2.1.1. Patrón arquitectónico en capas. 2.1.2. Aplicación del patrón arquitectónico en capas a los sistemas y servicios software. 2.1.3. Principios de diseño de la arquitectura en capas. 2.1.4. Patrón arquitectónico orientación a objetos. 2.1.5. Aplicación del patrón arquitectónico orientación a objetos a los sistemas software. 2.1.6. Principios de diseño de la arquitectura orientada a objetos.	Saber: - Identificar arquitectura en capas y orientada a objetos en el desarrollo de patrón arquitectónico en capas, aplicación arquitectónica en capas a los sistemas y servicios software. - Conocer los principios del diseño de la arquitectura en capas, patrón arquitectónico orientado a objetos, aplicación del patrón arquitectónico orientado a objetos a los sistemas software y principios de diseño de la arquitectura orientada a objetos en el contexto de la ingeniería de software.	- Preguntas intercaladas para evaluar los conocimientos previos. - Exposición por parte del profesor de material teórico. - Complementar información con material audiovisual. - Actividades para estructuración y apropiación de información: análisis de casos, resúmenes y/o mapas conceptuales.	Evaluación diagnóstica: - Rescatar conocimientos previos con preguntas intercaladas. Evaluación Formativa: - Análisis de caso de problemas resueltos. Instrumento de evaluación: - Lista de cotejo y/o rúbrica. Evaluación Sumativa: -Examen escrito. Instrumento de evaluación: - Cuestionario.	- Planteamiento y diseño del software de la arquitectura en capas y orientada a objetos de un problema laboral o cotidiano en el que se pueda aplicar patrón arquitectónico en capas, y de orientada a objetos, utilizando principios de diseño en capas y orientada a objetos y servicios de software.

Continuación: Tabla 4.2. Desglose específico de la unidad "Arquitectura en capas y orientada a objetos."				
Temas y subtemas (secuencia)	Criterios de desempeño	Estrategias didácticas	Estrategias de evaluación	Producto Integrador de la unidad
	Saber hacer: - Resolver ejercicios de la arquitectura en capas y orientada a objetos en el desarrollo de patrón arquitectónico en capas, aplicación arquitectónica en capas a los sistemas y servicios software, principios del diseño de la arquitectura en capas, patrón arquitectónico orientado a objetos, aplicación del patrón arquitectónico orientado a objetos a los sistemas software y principios de diseño de la arquitectura orientada a objetos en el contexto de la ingeniería de software. Ser: - Aporta puntos de vista con apertura a aprender de los otros y considera los de otras personas de manera reflexiva y respetuosa.			

Bibliografía

- Meszaros, G. (2007). refactoring test code. En XUnit test patterns(pp. 130-145). Boston: Pearson Education.
- Pressman, R.S.; Maxim, B.R. (2015). A practitioner's approach. En Software engineering (pp. 140-150). Estados Unidos de América: McGraw Hill.
- Sommerville, I. (2016). Software engineering. Estados Unidos de América: Pearson Education.

Sección IV. Desglose específico por cada unidad formativa

Tabla 4.3. Desglose específico de la unidad "Diseño del software utilizando metodologías clásicas."

Número y nombre de la unidad:	3. Diseño del software utilizando metodologías clásicas.					
Tiempo y porcentaje para esta unidad:	Teoría:	5 horas	Práctica:	10 horas	Porcentaje del programa:	16.67%
Aprendizajes esperados:	Conocer las metodologías de diseño del software, para aplicar la metodología adecuada en cada caso.					

Temas y subtemas (secuencia)	Criterios de desempeño	Estrategias didácticas	Estrategias de evaluación	Producto Integrador de la unidad (Evidencia de aprendizaje de la unidad)
3. Diseño del software utilizando metodologías clásicas. 3.1.1. Diseño del software utilizando metodologías clásicas. 3.1.2. Diseño de la capa de dominio. 3.1.3. Patrones Domain Model y Transaction Script. 3.1.4. Patrones de diseño de la capa de dominio. 3.1.5. Uso de servicios software. 3.1.6. Diseño de la capa de presentación. 3.1.7. Diseño externo de un sistema software. 3.1.8. Diseño interno de la capa de presentación. 3.1.9. Diseño de la capa de datos. 3.1.10. Tecnología de bases de datos relacionales. 3.1.11. Estrategias de gestión de la persistencia.	Saber: - Identificar el diseño del software utilizando metodologías clásicas, diseño de la capa de dominio, patrones: Domain Model y Transaction Script, - Conocer patrones de diseño de la capa de dominio, uso de servicios software, diseño de la capa de presentación, diseño extremo de un sistema de software, diseño interno de la capa de presentación, diseño de la capa de datos, tecnología de bases de datos relacionales, estrategias de gestión, generación automática y diseño directo de la	- Preguntas intercaladas para evaluar los conocimientos previos. - Exposición por parte del profesor de material teórico. - Complementar información con material audiovisual. - Actividades para estructuración y apropiación de información: análisis de caso, resúmenes y/o mapas conceptuales.	Evaluación diagnóstica: - Rescatar conocimientos previos con preguntas intercaladas. Evaluación Formativa: - Análisis de caso de problemas resueltos. Instrumento de evaluación: - Lista de cotejo y/o rúbrica. Evaluación Sumativa: -Examen escrito. Instrumento de evaluación: - Cuestionario.	- Planteamiento y diseño del software utilizando metodologías clásicas de un problema laboral o cotidiano en el que se pueda aplicar diseño de la capa de dominio, patrones: Domain Model y Transaction Script, patrones de diseño de la capa de dominio, uso de servicios software, diseño de la capa de presentación, diseño extremo de un sistema de software, diseño interno de la capa de presentación, diseño de la capa de datos, tecnología de bases de datos relacionales, estrategias de gestión, generación automática y diseño directo

Continuación: Tabla 4.3. Desglose específico de la unidad "Diseño del software utilizando metodologías clásicas."				
Temas y subtemas (secuencia)	Criterios de desempeño	Estrategias didácticas	Estrategias de evaluación	Producto Integrador de la unidad
3.1.12. Generación automática de persistencia. 3.1.13. Diseño directo de la persistencia.	persistencia en el contexto de la ingeniería de software. Saber hacer: - Resolver ejercicios de diseño de software utilizando metodologías clásicas en el desarrollo de diseño de capa de dominio, patrones Domain Model y Transaction Script, patrones de diseño de la capa de dominio, uso de servicios software, diseño de la capa de presentación, diseño extremo de un sistema de software, diseño interno de la capa de presentación, diseño de la capa de datos, tecnología de bases de datos relacionales, estrategias de gestión, generación automática y diseño directo de la persistencia en el contexto de la ingeniería de software. Ser: - Aporta puntos de vista con apertura a			de la persistencia.

Continuación: Tabla 4.3. Desglose específico de la unidad "Diseño del software utilizando metodologías clásicas."				
Temas y subtemas (secuencia)	Criterios de desempeño	Estrategias didácticas	Estrategias de evaluación	Producto Integrador de la unidad
	aprender de los otros y considera los de otras personas de manera reflexiva y respetuosa.			
Bibliografía				
- Meszaros, G. (2007). refactoring test code. En XUnit test patterns(pp. 130-145). Boston: Pearson Education. - Pressman, R.S.; Maxim, B.R. (2015). A practitioner's approach. En Software engineering (pp. 140-150). Estados Unidos de América: McGraw Hill. - Sommerville, I. (2016). Software engineering. Estados Unidos de América: Pearson Education.				

Sección IV. Desglose específico por cada unidad formativa

Tabla 4.4. Desglose específico de la unidad "Diseño de software utilizando metodologías ágiles."

Número y nombre de la unidad:	4. Diseño de software utilizando metodologías ágiles.					
Tiempo y porcentaje para esta unidad:	Teoría:	5 horas	Práctica:	14 horas	Porcentaje del programa:	21.11%
Aprendizajes esperados:	- Conocer el diseño de software utilizando metodologías ágiles: Extreme Programming. TDD. Test en TDD, Coding en TDD, Pair Programming, Simple Design and Incremental design and architecture, Code Smells and Refactorings para diseñar software utilizando metodologías ágiles.					

Temas y subtemas (secuencia)	**Criterios de desempeño**	**Estrategias didácticas**	**Estrategias de evaluación**	**Producto Integrador de la unidad (Evidencia de aprendizaje de la unidad)**
4. Diseño de software utilizando metodologías ágiles. 4.1.1. Diseño del software utilizando metodologías ágiles. Extreme Programming. TDD. Tests en TDD. 4.1.2. Coding en TDD. Pair Programming. Simple Design and Incremental design and architecture. Code Smells and Refactorings.	Saber: - Identificar el diseño de software utilizando metodologías ágiles: Extreme Programming. TDD. Test en TDD, Coding en TDD, Pair Programming, Simple Design and Incremental design and architecture, Code Smells and Refactorings. Saber hacer: - Resolver ejercicios de diseño de software utilizando metodologías ágiles enel desarrollo de Extreme Programming. TDD. Test en TDD, Coding en TDD, PairProgramming, Simple Design and Incremental design and architecture, Code Smells and Refactorings en el	- Preguntas intercaladas para evaluar los conocimientos previos. - Exposición por parte del profesor de material teórico. - Complementar información con material audiovisual. - Actividades para estructuración y apropiación de información: análisis de caso, resúmenes y/o mapas conceptuales.	Evaluación diagnóstica: - Rescatar conocimientos previos con preguntas intercaladas. Evaluación Formativa: - Análisis de caso de problemas resueltos. Instrumento de evaluación: - Lista de cotejo y/o rúbrica. Evaluación Sumativa: -Examen escrito. Instrumento de evaluación: - Cuestionario.	- Planteamiento y diseño del software utilizando metodologías ágiles de un problema laboral o cotidiano en el que se pueda aplicar Extreme Programming. TDD. Test en TDD, Coding en TDD, Pair Programming, Simple Design and Incremental design and architecture, Code Smells and Refactorings.

Continuación: Tabla 4.4. Desglose específico de la unidad "Diseño de software utilizando metodologías ágiles."				
Temas y subtemas (secuencia)	Criterios de desempeño	Estrategias didácticas	Estrategias de evaluación	Producto Integrador de la unidad
	contexto de la ingeniería de software. Ser: - Aporta puntos de vista con apertura a aprender de los otros y considera los de otras personas de manera reflexiva y respetuosa.			

Bibliografía

- Meszaros, G. (2007). refactoring test code. En XUnit test patterns(pp. 130-145). Boston: Pearson Education.
- Pressman, R.S.; Maxim, B.R. (2015). A practitioner's approach. En Software engineering (pp. 140-150). Estados Unidos de América: McGraw Hill.
- Sommerville, I. (2016). Software engineering. Estados Unidos de América: Pearson Education.

V. Perfil docente

Tabla 5. Descripción del perfil docente

Perfil deseable docente para impartir la asignatura
Carrera(s): Ingeniero en Sistemas, titulado o carrera a fin o maestría relacionada con el área de conocimiento o carrera afín - Con experiencia docente o en el campo. - Experiencia mínima de dos años - Manejo de TIC´s. Con habilidades pedagógicas y uso de metodologías alternativas de enseñanza.