CENTRO UNIVERSITARIO DE TECNOLOGÍA Y ARTE DIGITAL

PLANIFICACIÓN DE LA DOCENCIA UNIVERSITARIA

GUÍA DOCENTE

PROGRAMACIÓN AVANZADA

1. DATOS DE IDENTIFICACIÓN DE LA ASIGNATURA.

Título:				**Grado en Ingeniería en Desarrollo de Contenidos Digitales**	
Facultad:				**Centro Universitario de Tecnología y Arte Digital (U-tad)**	
Departamento/Instituto:					
Materia:				**Fundamentos de Ingeniería del Software**	
Denominación de la asignatura:				**Técnicas avanzadas de programación**	
Código:				**0048044**	
Curso:				**Cuarto**	
Semestre:				**Segundo**	
Tipo de asignatura (básica, obligatoria u optativa):				**Optativa**	
Créditos ECTS:				**6**	
Modalidad/es de enseñanza:				**Presencial**	
Lengua vehicular:				**Español**	
Equipo docente:				**Tomás Fernández**	
Profesor/a:				**Tomás Fernández**	
Grupos:				**IDCD4**	
Despacho:				**Sala de profesores**	
Teléfono:	91 6402811	Ext.	113	E-mail:	tomas.fernandez@live.u-tad.com
Página web: http://u-tad.blackboard.com					

2. REQUISITOS PREVIOS.

Esenciales:
Haber cursado las asignaturas de Introducción a la Programación, Algoritmos y Estructuras de Datos y Programación a Bajo Nivel
Aconsejables:
Haber aprobado las asignaturas de Introducción a la Programación, Algoritmos y Estructuras de Datos y Programación a Bajo Nivel

3. SENTIDO Y APORTACIONES DE LA ASIGNATURA AL PLAN DE ESTUDIOS.

Campo de conocimiento al que pertenece la asignatura.
Esta asignatura pertenece al Módulo de Fundamentos, a la Materia de Fundamentos de la Ingeniería del Software
Relación de interdisciplinariedad con otras asignaturas del curriculum.
Esta asignatura entronca verticalmente con Introducción a la Programación, Algoritmos y Estructuras de Datos y Programación a Bajo Nivel, suponiendo la culminación de las asignaturas de programación orientada a objetos.
Aportaciones al plan de estudios e interés profesional de la asignatura.
Aporta el aprendizaje de conocimientos avanzados de programación orientada a objetos.

4. RESULTADOS DE APRENDIZAJE EN RELACIÓN CON LAS COMPETENCIAS QUE DESARROLLA LA ASIGNATURA.

COMPETENCIAS ESPECÍFICAS	RESULTADOS DE APRENDIZAJE RELACIONADOS CON LAS COMPETENCIAS ESPECÍFICAS
CE4 - Tener conocimiento de la estructura, arquitectura, organización, funcionamiento e interconexión de los sistemas informáticos y los fundamentos de su programación CE6 - Poseer capacidad para diseñar, desarrollar, seleccionar y evaluar aplicaciones y sistemas informáticos, asegurando su fiabilidad, seguridad y calidad CE9 - Mostrar conocimiento, diseño y aplicación de los procedimientos algorítmicos, tipos y estructuras de datos básicos de las tecnologías informáticas para diseñar soluciones a problemas, analizando la idoneidad y complejidad de los algoritmos propuestos CE11 - Tener conocimiento de las características, funcionalidades y estructura de los Sistemas Operativos y diseñar e implementar aplicaciones basadas en sus servicios CE19 - Adquirir los fundamentos de las diversas ramas de especialización relacionadas con el área del desarrollo de contenidos digitales y software.	Comprender los elementos de lenguajes de programación de distintos paradigmas. Identificar las principales estructuras de datos y técnicas algorítmicas y sus complejidades. Valorar el uso de una u otra técnica o estructura de datos según el problema a resolver. Adquirir conocimientos sólidos de un lenguaje de bajo nivel como C y la gestión de memoria explícita. Programar sistemas que requieran un conocimiento del funcionamiento del software en las capas de más bajo nivel. Conocer técnicas avanzadas de programación empleadas en proyectos software profesionales y de dimensión y complejidad considerable.

5. CONTENIDOS / TEMARIO / UNIDADES DIDÁCTICAS

Módulo 1: Programación modular.
- Tema 1: Repaso y profundización en las herramientas: clases, interfaces y plantillas.
- Tema 2: Componentes y programación guiada por datos.
- Tema 3: Librerías de enlace dinámico y “Plugins”.

Módulo 2: Depuración y optimización.
- Tema 4: Detección y depuración de fugas de memoria.
- Tema 5: Detección y optimización cuellos de botella.

Módulo 3: Programación multiplataforma.
- Tema 6: Organización de un desarrollo multiplataforma.
- Tema 7: Diseño de arquitectura independiente de plataforma.
- Tema 8: Gestión de las especificidades de cada plataforma.

Módulo 4: Programación en red.
- Tema 9: Patrones utilizados en la programación en red.
- Tema 10: Juegos en red.
- Tema 11: Servidores de datos.
- Tema 12: Servidores de partidas.

Módulo 5: Desarrollos con varios lenguajes de programación.
- Tema 13: Interés de desarrollar con varios lenguajes un mismo sistema
- Tema 14: Integración de C++ con un leguaje de scripting.

6. CRONOGRAMA

UNIDADES DIDÁCTICAS / TEMAS	PERÍODO TEMPORAL
Módulo 1	Semana 1, 2, 3, 4
Módulo 2	Semana 5 y 6
Módulo 3	Semana 7, 8, 9 y 10
Módulo 4	Semana 11, 12, 13 y 14
Módulo 5	Semana 15

7. MODALIDADES ORGANIZATIVAS Y MÉTODOS DE ENSEÑANZA

MODALIDAD ORGANIZATIVA	MÉTODO DE ENSEÑANZA	COMPETENCIAS RELACIONADAS	HORAS PRESENCIALES	TRABAJO AUTÓNOMO	TOTAL DE HORAS
Clases teóricas	Lección magistral	CE4, CE6, CE9, CE11, CE19	21	1	22
Seminarios y talleres	Estudio de casos Resolución de ejercicios y problemas		0	0	0
Clases prácticas	Aprendizaje basado en problemas Aprendizaje orientado a proyectos	CE4, CE6, CE9, CE11, CE19	22	0	22
Practicas externas			0	0	0
Tutorías	Aprendizaje orientado a proyectos Aprendizaje basado en problemas	CE4, CE6, CE9, CE11, CE19	7	0	7
Actividades de evaluación		CE4, CE6, CE9, CE11, CE19	7	0	7
Estudio y trabajo en grupo	Aprendizaje cooperativo	CE4, CE6, CE9, CE11, CE19	1	21	23
Estudio y trabajo autónomo, individual	Estudio de casos Resolución de ejercicios y problemas Aprendizaje basado en problemas Aprendizaje orientado a proyectos	CE4, CE6, CE9, CE11, CE19	0	68	68

Durante la asignatura los alumnos deberán desarrollar un juego multijugador y multiplataforma donde poner en práctica los conocimientos teóricos adquiridos.
Las clases teórico-prácticas seguirán el siguiente esquema:

1. Presentación teórica del tema al que se dedica la clase ilustrada mediante ejemplos y abierta a la participación del alumnado para preguntas, comentarios o aclaraciones.
2. Tiempo dedicado al desarrollo del juego donde se pondrán en práctica los conocimientos explicados.

Las clases serán lo más interactivas posible, siempre abierta a la participación y resolución de inquietudes del alumnado en cualquier momento de la clase.

8. SISTEMA DE EVALUACIÓN

ACTIVIDAD DE EVALUACIÓN	CRITERIOS DE EVALUACIÓN	VALORACIÓN RESPECTO A LA CALIFICACIÓN FINAL (%)
Evaluación continua: Programación de las distintas partes del proyecto de la asignatura.	Cada parte del proyecto se evaluará con una nota de 0 a 10. Siendo la nota final, la media de las notas de las distintas partes.	50%
Examen final ordinario	De 0 a 10. Es obligatorio obtener al menos un 5 para aprobar la asignatura.	50%
Examen final extraordinario	De 0 a 10. Es obligatorio obtener al menos un 5 para aprobar la asignatura.	50%

Consideraciones generales acerca de la evaluación:

- Es necesario obtener al menos un 5 en la nota final para poder aprobar la asignatura. Existen dos oportunidades para ello: la convocatoria ordinaria y la extraordinaria.
- La nota de la práctica sirve tanto para la convocatoria ordinaria como para la extraordinaria, en caso de necesitarse.
- El porcentaje de presencialidad es del 80%.
- Las notas del examen final de la práctica no se guardan entre cursos académicos sucesivos.
- La realización de la práctica puede ser individual o por parejas, por lo cual la asignatura COMPLETA estará suspensa si se descubre que un alumno o una pareja ha copiado a otro alumno o pareja (ambos estarán suspensos). Además, la universidad abrirá expedientes disciplinarios a ambos alumnos, pudiendo desembocar incluso en su expulsión.

Los exámenes serán de tipo test, sumando cada respuesta acertada y restando cada respuesta incorrecta.

La práctica en C/C++ que el alumno realice deberán poseer las siguientes características:

1. Debe cumplir las especificaciones expuestas.
2. Debe ser clara y bien organizada, no sólo es suficiente que funcione.
3. Debe funcionar correctamente sin errores que impidan su utilización.
4. Se valorará muy positivamente que esté bien documentada en el código.

Cualquier escrito que el alumno presente (problemas, exámenes, comentarios de los programas, etc.) deberá estar bien presentado, correctamente redactado (con las

comas, puntos y puntos y aparte en su lugar adecuado) y sin faltas ortográficas. La nota del escrito podrá bajar hasta un 20% en caso contrario, ya que a un universitario se le exige calidad máxima en su expresión escrita.

9. BIBLIOGRAFÍA / WEBGRAFÍA

Bibliografía general

Bibliografía básica:

- Bjarne Stroustrup, **El lenguaje de programación C++.** Addison-Wesley. ISBN-13: 978-8478290468

Bibliografía de ampliación:

- Herb Sutter, **More Exceptional C++: 40 New Engineering Puzzles, Programming Problems, and Solutions**. Addison-Wesley Professional. ISBN-13: 978-0201704341
- Meyers, **Effective C++: 55 Specific Ways to Improve Your Programs and Designs**. Addison-Wesley Professional. ISBN-13: 978-0321334879

Sitios web de interés:

- http://www.gamedev.net/
- http://gafferongames.com/
- http://www.gamasutra.com/
- http://gameprogrammingpatterns.com/

Bibliografía recomendada por temas

Toda la bibliografía básica sirve para todos los temas.

10.- OBSERVACIONES

Recursos necesarios para la docencia:

- Ordenador del profesor conectado a proyector. Software instalado: “Microsoft PowerPoint”, “Acrobat Reader”, “Microsoft Visual C++ 2013”, VMWare Workstation o VirtualBox.
- Pizarra digital
- Pizarra blanca
- Un ordenador para cada alumno en el aula. Software instalado: “Microsoft Visual C++ 2013 Express Edition” y “Acrobat Reader”.

Recursos necesarios para el trabajo personal del alumno, fuera de clase:

- Ordenador de gama doméstica
- Conexión a internet
- Software “Microsoft Visual C++ 2013”
- Software “Acrobat Reader”