COMILLAS
UNIVERSIDAD PONTIFICIA
ICAI ICADE CIHS

GUÍA DOCENTE
2023 - 2024

FICHA TÉCNICA DE LA ASIGNATURA

Datos de la asignatura	
Nombre completo	Paradigmas y técnicas de programación
Código	DTC-IMAT-315
Título	Grado en Ingeniería Matemática e Inteligencia Artificial
Impartido en	Grado en Ingeniería Matemática e Inteligencia Artificial [Tercer Curso]
Nivel	Reglada Grado Europeo
Cuatrimestre	Semestral
Créditos	6,0 ECTS
Carácter	Obligatoria (Grado)
Departamento / Área	Departamento de Telemática y Computación
Responsable	Arturo Serrano Martínez, Daniel Morell Mañas
Horario	Consultar horarios de la escuela
Horario de tutorías	Se comunicará el primer día de clase

Datos del profesorado	
Profesor	
Nombre	David Contreras Bárcena
Departamento / Área	Departamento de Telemática y Computación
Despacho	Alberto Aguilera 25
Correo electrónico	davidcb@comillas.edu
Teléfono	4235
Profesor	
Nombre	Arturo Serrano Martínez
Departamento / Área	Departamento de Telemática y Computación
Correo electrónico	asmartinez@icai.comillas.edu
Profesor	
Nombre	Daniel Morell Mañas
Departamento / Área	Departamento de Telemática y Computación
Correo electrónico	dmorell@icai.comillas.edu

DATOS ESPECÍFICOS DE LA ASIGNATURA

Contextualización de la asignatura

Aportación al perfil profesional de la titulación

El objetivo principal de esta asignatura es proporcionar a los estudiantes una comprensión profunda de los diferentes paradigmas de

programación, centrándose especialmente en el paradigma de la Programación Orientada a Objetos (POO).

A lo largo del curso, los estudiantes explorarán los conceptos teóricos y las prácticas de programación necesarios para comprender y aplicar eficazmente el paradigma de la POO. Se hará hincapié en el lenguaje C# como herramienta principal para el aprendizaje y la implementación de la POO, y a su vez trasladar dichos conocimientos a Python.

El curso comenzará con una introducción a los paradigmas de programación, donde se explorarán las diferentes formas de abordar la resolución de problemas y se discutirán las ventajas y desventajas de cada paradigma. A continuación, se profundizará en el paradigma de la POO, tratando aspectos clave como objetos y clases, herencia, polimorfismo y encapsulación.

Los estudiantes aprenderán a diseñar, implementar y refactorizar código utilizando los principios de la POO, incluyendo los principios SOLID y Clean Code. Se explorarán las clases básicas del lenguaje C# y las estructuras de datos fundamentales, así como la manipulación de archivos y el uso de API.

Además, la asignatura abordará temas avanzados, como los patrones de diseño, que permitirán a los estudiantes desarrollar soluciones de software eficientes y escalables. Se introducirá el uso de Unity, una plataforma popular para el desarrollo de videojuegos, donde se aplicarán los conceptos de la POO y se explorará la integración de agentes ML dentro de un videojuego.

También, se abordará la programación con hilos (threads) y sockets, permitiendo a los estudiantes diseñar aplicaciones concurrentes y comunicarse a través de redes. Por último, se presentará una introducción a la Programación Funcional en Python, explorando conceptos como funciones de orden superior y recursión, ampliando así las habilidades de los estudiantes en diferentes paradigmas de programación.

La asignatura también incluirá una parte práctica y de proyecto, donde los estudiantes trabajarán en parejas para crear y desarrollar un videojuego a lo largo del curso. Este proyecto permitirá a los estudiantes aplicar los conceptos aprendidos y fortalecer sus habilidades de programación y trabajo en equipo.

Al finalizar la asignatura, los estudiantes habrán adquirido un sólido conocimiento de la Programación Orientada a Objetos y estarán preparados para abordar proyectos de desarrollo de software más complejos. Además, habrán desarrollado habilidades prácticas y críticas para diseñar y construir soluciones eficientes y mantenibles utilizando los principios de la POO.

Prerequisitos

- Competencia básica en programación.

Competencias - Objetivos

Competencias

GENERALES	
CG04	Conocimientos básicos sobre el uso y programación de los ordenadores, sistemas operativos, bases de datos y programas informáticos con aplicación en ingeniería.
CG05	Conocimiento de la estructura, organización, funcionamiento e interconexión de los sistemas informáticos, los fundamentos de su programación, y su aplicación para la resolución de problemas propios de la ingeniería
ESPECÍFICAS	
CE10	Conocimiento de la sintaxis, las estructuras principales y los elementos básicos de un lenguaje de programación en el contexto del análisis de datos y la inteligencia artificial

CE11	Dominio de las principales estructuras de datos y técnicas algorítmicas, siendo capaz de implementarlas en distintos lenguajes de programación conociendo su complejidad computacional
CE12	Conocimiento de los fundamentos y beneficios de los distintos paradigmas de programación para saber aplicarlos en cada problema particular para maximizar su eficiencia computacional y distinguir la diferencia que existe entre los lenguajes de programación nativos e interpretados.
CE17	Capacidad para analizar y distinguir las características, funcionalidades y estructura de los sistemas operativos y diseñar aplicaciones basadas en sus servicios.

Resultados de Aprendizaje	
RA1	Conocer los paradigmas de programación y ser capaz de elegir el mejor para cada tipo de problema
RA2	Dominar las características más importantes de la programación orientada a objetos
RA3	Conocer los diagramas básicos del lenguaje de modelado UML para el análisis y diseño de un programa orientado a objetos
RA4	Conocer el funcionamiento de una API de un lenguaje de programación y dominar la manipulación de sus objetos
RA5	Implementar programas básicos de paralelización de código basadas en hilos de ejecución (threads)
RA6	Implementar programas básicos cliente/servidor basados en sockets TCP
RA7	Conocer las características más importantes de la programación funcional

BLOQUES TEMÁTICOS Y CONTENIDOS

Contenidos – Bloques Temáticos

1. Introducción a los paradigmas de programación

- Definición de paradigmas de programación.
- Principales paradigmas: imperativo, orientado a objetos, funcional, lógico, etc.
- Ventajas y desventajas de cada paradigma.
- Ejemplos prácticos de aplicaciones en diferentes paradigmas.

2. Introducción a Unity

- Conceptos básicos de Unity.
- Creación de escenas y objetos en Unity.
- Manipulación de componentes y scripts.

3. Programación Orientada a Objetos utilizando C#

- Conceptos básicos de la programación orientada a objetos (POO).
- Objetos y clases.
- Herencia y polimorfismo.

- Aplicación de la POO en la resolución de problemas.

4. Introducción a la arquitectura del software

- Principios y conceptos de la arquitectura del software.
- Análisis de software.
- Diseño de software orientado a objetos.
- Componentes y conectores en la arquitectura.
- Patrones arquitectónicos comunes.

5. Clases básicas del lenguaje y API.

- Clases y estructuras básicas.
- Estructuras de datos básicas: listas, pilas, colas, etc.
- Manipulación de ficheros y acceso a bases de datos.
- Uso de API en aplicaciones.

6. Patrones de diseño, principios SOLID y Clean Code

- Principios SOLID (Single Responsibility, Open/Closed, Liskov Substitution, Interface Segregation, Dependency Inversion).
- Buenas prácticas de programación y clean code.
- Introducción a los patrones de diseño.

7. Agentes ML en Unity

- Introducción a la inteligencia artificial y el aprendizaje automático.
- Creación de agentes de IA en Unity utilizando herramientas de ML.

8. Introducción a la Programación Funcional en Python

- Conceptos básicos de la programación funcional.
- Sintaxis y características de la programación funcional en Python.
- Aplicación de la programación funcional en la resolución de problemas.

9. Introducción al Diseño e implementación de aplicaciones basadas en hilos (Threads) y a la Programación con Sockets.

- Conceptos básicos de hilos (threads) y concurrencia.
- Diseño e implementación de aplicaciones multihilo.
- Introducción a la programación con sockets para comunicación en red.

METODOLOGÍA DOCENTE

Aspectos metodológicos generales de la asignatura

El método de evaluación para la asignatura de Paradigmas de Programación se basa en una evaluación continua que integra diferentes aspectos del aprendizaje. Se valorará tanto el conocimiento teórico como la aplicación práctica de los conceptos, fomentando la participación de los estudiantes. Con un peso significativo en la evaluación, los exámenes final e intermedio evaluarán la comprensión de los contenidos teóricos. Además, se tomará en cuenta la participación en clase y la colaboración en las prácticas, donde los estudiantes desarrollarán un videojuego progresivamente. También cabe destacar el hincapié en el trabajo personal y en equipo del alumno fuera del aula. Esta estructura de evaluación busca promover un aprendizaje integral y fortalecer las habilidades tanto individuales como de trabajo en equipo.

Las metodologías docentes a seguir en estas actividades serán:

- Lección magistral
- Aprendizaje práctico
- Aprendizaje colaborativo
- Clase invertida

Metodología Presencial: Actividades

Las actividades formativas se desarrollarán durante las 4 horas de clase a la semana que se distribuirán secuencialmente de la siguiente forma:

- **2h Teoría + 2h Práctica individual y colaborativa**

Estas sesiones estarán compuestas por:

- **Clases magistrales expositivas y participativas:**
  - Sesiones de 2h de Teoría.
  - El profesor realizará un exposición de los contenidos teóricos, combinando la clase magistral con la programación en directo (livecoding) de pequeños ejemplos ilustrativos.
  - Los códigos generados en el aula estarán a disposición del alumno en el repositorio de la asignatura.
- **Ejercicios prácticos y resolución de problemas:**
  - El alumno planteará dudas sobre los conceptos aprendidos tanto en las sesiones teóricas como en las prácticas o en el desarrollo del proyecto.
  - Se crearán dinámicas interacticas de resolución de las dudas entre toda la clase con programación de ejemplos por parte del profesor y los alumnos de forma colaborativa.
- **Sesiones Prácticas con uso de software:**
  - Las prácticas de la asignatura se dividirán en dos fases. En la fase inicial, que abarcará las primeras 5 semanas, se llevarán a cabo prácticas sencillas con el objetivo de afianzar el entendimiento de los conceptos teóricos impartidos en clase. A partir de la sexta semana, comenzará una práctica incremental en parejas, donde se fomentará el trabajo en equipo y la colaboración entre los estudiantes. El objetivo será crear y desarrollar un videojuego a lo largo del curso, agregando contenido de forma progresiva semana tras semana. Este enfoque práctico permitirá a los estudiantes aplicar los conceptos y técnicas aprendidos de manera tangible, fortaleciendo sus habilidades de programación y trabajo en equipo.
  - **Proceso de desarrollo del videojuego:**
    1. **Selección del videojuego:** A partir de la semana 6, cada pareja deberá seleccionar un género o estilo de videojuego que deseen desarrollar. Puede ser un juego de plataformas, acción, aventura, puzles, entre otros. Esta elección debe ser comunicada al profesor para su aprobación y seguimiento.
    2. **Diseño inicial**: Una vez seleccionado el género del videojuego, cada pareja deberá elaborar un diseño inicial que contemple los aspectos básicos del juego, como la mecánica principal, los personajes, los niveles o escenarios, entre otros. Este diseño servirá como punto de partida para el desarrollo progresivo del videojuego.
    3. **Incremento semanal**: A partir de la segunda semana, cada pareja irá implementando y añadiendo contenido a su videojuego de forma semanal. En cada práctica, se establecerán objetivos específicos a alcanzar y nuevas funcionalidades a incorporar en el juego. Estos objetivos pueden incluir el desarrollo de nuevos niveles, la implementación de mecánicas adicionales, la mejora de la interfaz de usuario, entre otros.
    4. **Sesiones de seguimiento**: Durante las prácticas, se llevarán a cabo sesiones de seguimiento en las que el profesor revisará el progreso de cada pareja. Estas sesiones

CE11, CE10, CG04, CE12, CG05, CE17

servirán para brindar retroalimentación, resolver dudas y proporcionar orientación adicional en el desarrollo del videojuego.

5. **Presentación y evaluación**: Al finalizar el curso, cada pareja deberá presentar el videojuego desarrollado en una sesión específica. Durante esta presentación, se evaluará tanto la funcionalidad del juego como la calidad del código, la implementación de los conceptos aprendidos en clase y la originalidad en el diseño.

- **Aspectos a tener en cuenta:**
  - **Calidad de código:** Se valorará la estructura y organización del código, su eficiencia y optimización, así como su sostenibilidad y legibilidad.
  - **Documentación:** Cada pareja deberá mantener una documentación actualizada del proceso de desarrollo, incluyendo el diseño inicial, los cambios realizados y los desafíos encontrados. Esta documentación será requerida al final del curso para su revisión.
  - **Equidad en la colaboración:** Es importante que ambas personas de la pareja participen activamente en el desarrollo del videojuego. Se recomienda establecer mecanismos de trabajo colaborativo y equitativo para garantizar la participación y el aprendizaje de todos los integrantes.
  - **Retroalimentación entre parejas:** Se fomenta la retroalimentación constructiva entre las parejas, lo que permitirá compartir conocimientos, resolver problemas en conjunto y mejorar la calidad de los videojuegos desarrollados.
- Este enfoque de prácticas basado en el desarrollo progresivo de un videojuego en parejas permitirá a los estudiantes aplicar de manera práctica los conocimientos adquiridos, fortalecer sus habilidades de programación y trabajar en equipo, aspectos fundamentales en el ámbito de la ingeniería informática.

- **Actividades de evaluación continua del rendimiento:**
  - Se realizarán pruebas, desarrollarán prácticas complementarias a las semanales y retos gamificados.
  - Test de autoevaluación del tema: el alumno identifica si domina los conceptos teóricos.
- **Tutoría para la resolución de dudas:** esta actividad se realizará de forma implícita durante el resto de actividades descritas.

Metodología No presencial: Actividades

- **Ejercicios prácticos y resolución de problemas:**
  - El alumno recibirá de forma periódica tareas a realizar entre las sesiones presenciales. Estas tareas reflejarán los conocimientos adquiridos por el alumno entre sesión y sesión.
  - Dichos ejercicios tendrán que ser enviados para su evaluación antes de la fecha especificada por el profesor. Posteriormente, se subirá la solución a la plataforma de enseñanza.
- **Sesiones prácticas con uso de software:** Esto incluye el trabajo fuera del aula tanto en ejercicios individuales, como el desarrollo individual y por parejas del proyecto final.
  - Al finalizar una sesión teórica, el alumno trabajará fuera del aula en la sesión práctica posterior. Dicha práctica pondrá a prueba los conocimientos del alumno sobre lo aprendido en la sesión teórica. El alumno tendrá que llegar a la sesión práctica con el 80% de la misma desarrollada. Esto variará en función de la complejidad de dicha práctica.
  - Una vez comience el proyecto final, el alumno tendrá que desarrollar progresivamente y de forma equitativa dicho proyecto con su compañero fuera del aula.
    - Se valorarán los criterios indicados en la metodología presencial.
    - Será necesario demostrar la colaboración de los compañeros en el proyecto final (control de versiones o similar).
  - Se valorará negativamente el uso de herramientas o código de terceros siempre que el alumno no sepa justificar la comprensión del mismo.
- **Estudio personal:** El objetivo principal del trabajo no presencial es llegar a entender y comprender

CE12, CG04, CE11, CE17, CG05, CE10

los conceptos teóricos de la asignatura, así como ser capaz de poner en práctica estos conocimientos para resolver los diferentes tipos de problemas. Después de cada explicación teórica el profesor subirá a la web todos los códigos desarrollados y el alumno deberá revisarlos y plantearse cuestiones "Whatif" para asimilar mejor los conceptos teóricos. La forma de estudiar la asignatura será la siguiente:

- Estudiar los conceptos explicados con una papel y un lápiz, sin necesidad de trabajar con el ordenador. El alumno debe saber analizar, diseñar y resolver problemas de un forma abstracta sin el ordenador.
- Analizará el código suministrado por el profesor con el fin de asentar los coneptos teóricos.
- Por último, deberá ser capaz de programar los ejercicios realizados por el profesor desde cero: sin copiar y pegar ni mirar los códigos suministrados.

RESUMEN HORAS DE TRABAJO DEL ALUMNO

HORAS PRESENCIALES				
Clases magistrales expositivas y participativas	Sesiones prácticas con uso de software	Tutorías para resolución de dudas	Ejercicios prácticos y resolución de problemas	Actividades de evaluación continua del rendimiento
36.00	16.00	5.00	4.00	4.00
HORAS NO PRESENCIALES				
Estudio personal	Ejercicios prácticos y resolución de problemas	Sesiones prácticas con uso de software		
45.00	22.00	48.00		
CRÉDITOS ECTS: 6,0 (180,00 horas)				

EVALUACIÓN Y CRITERIOS DE CALIFICACIÓN

Actividades de evaluación	Criterios de evaluación	Peso

Exámenes:
• Prueba Intersemestral.
• Examen final.
	• **Prueba Intersemestral (15%):**
	  ◦ Su objetivo es evaluar la progresión de los estudiantes hasta ese punto y brindarles una retroalimentación temprana sobre su rendimiento.
	  ◦ Se evaluarán los conocimientos aprendidos hasta dicha fecha en la asignatura (Paradigmas, Conceptos y uso de POO, uso de API, conocimientos de Unity).
	• **Examen final (45%):**
	  ◦ Se evaluará la totalidad del contenido de la asignatura, incluyendo el temario de la prueba intersemestral.
		60 %

	Proyecto final que se entregará por parejas al	

Proyecto final	finalizar el curso. • Se evaluará la calidad de las implementaciones, la aplicación de los conceptos aprendidos, la organización del código y la capacidad de trabajar en equipo. • También se tomará en cuenta la documentación y la presentación final del videojuego.	30 %
Sesiones prácticas: • Retos colaborativos • Trabajos no presenciales • Prácticas	La actitud, participación y realización de las prácticas semanales y los retos planteados en sesiones colaborativas e individuales.	10 %

Calificaciones

Evaluación:

- 45% examen final
- 15% examen intermedio
- 10% prácticas
- 30% proyecto final

Notas:

Para poder aplicar la ponderación es imprescindible una nota mínima en el examen de 5.

La inasistencia al 15% o más de las horas presenciales de esta asignatura puede tener como consecuencia la imposibilidad de presentarse a las convocatorias ordinaria y extraordinaria.

BIBLIOGRAFÍA Y RECURSOS

Bibliografía Básica

Libro guía de la asignatura: Unity Artificial Intelligence Programming - Fourth Edition. Dr. Davide Aversa , Aung Sithu Kyaw , Clifford Peters.

Fundamentos Clean Code: Clean Code: A Handbook of Agile Software Craftsmanship 1st Edition. Robert C. Martin.

https://servicios.upcomillas.es/sedeelectronica/inicio.aspx?csv=02E4557CAA66F4A81663AD10CED66792