GUÍA DOCENTE DE PROYECTO DE ANALISIS Y DISEÑO DE SOFTWARE

La presente guía docente corresponde a la asignatura Proyecto de Análisis y Diseño de Software (PADS), aprobada para el curso lectivo 2016-2017 en Junta de Centro y publicada en su versión definitiva en la página web de la Escuela Politécnica Superior. El presente documento es una guía, y por tanto es posible que su desarrollo durante el curso sufra alguna variación.

ASIGNATURA

PROYECTO DE ANÁLISIS Y DISEÑO DE SOFTWARE (PADS)

1.1. Código

17833 del Grado en Ingeniería Informática

1.2. Materia

Análisis y Diseño de Software

1.3. Tipo

Formación obligatoria

1.4. Nivel

Grado

1.5. Curso

2º

1.6. Semestre

2º

1.7. Número de créditos

6 créditos ECTS

1.8. Requisitos previos

Esta asignatura asume conocimientos de programación estructurada por parte del estudiante, que deben haber sido adquiridos cursando las materias del módulo de *Programación y Estructuras de Datos* del plan de estudios. Es muy aconsejable haber superado con éxito las asignaturas *Programación I*, *Programación II* y *Proyecto de Programación* para un aprovechamiento de la asignatura PADS. La materia Análisis y Diseño de Software se desglosa en dos asignaturas: Análisis y Diseño de Software y Proyecto de Análisis y Diseño de Software, ambas impartidas en el 2º cuatrimestre

del segundo curso. Ya que ambas proporcionan formación complementaria y conceptos relacionados, **se recomienda cursarlas a la vez**.

Para garantizar la asimilación de los contenidos y la adquisición de habilidades se recomienda la lectura crítica de los textos de la bibliografía, el uso del material electrónico de esta asignatura disponible en la plataforma Moodle (https://moodle.uam.es) y la búsqueda activa de material complementario en la red. Es recomendable disponer de un dominio del inglés que permita al alumno leer la bibliografía de consulta. Asimismo, se requiere iniciativa personal y constancia para el diseño e implementación de aplicaciones, así como predisposición y empatía para el trabajo colaborativo en grupo.

1.9. Requisitos mínimos de asistencia a las sesiones presenciales

Los estudiantes que abandonen el método de evaluación continua, o no cumplan los requisitos extra, serán evaluados directamente mediante método de evaluación final (con ponderaciones diferentes a las de quienes se mantengan dentro de la evaluación continua), sin que ello les exima de realizar todas y cada una de las entregas parciales del proyecto en las mismas fechas establecidas (comunes a los estudiantes en evaluación continua).

ITINERARIO CON ASISTENCIA OBLIGATORIA A CLASE
La asistencia es obligatoria al menos en un 85%.

ITINERARIO SIN ASISTENCIA OBLIGATORIA A CLASE
La asistencia es muy recomendable aunque no obligatoria.

1.10. Datos del equipo docente

Coordinador:

Dr. Eduardo Pérez Pérez
Departamento de Ingeniería Informática
Escuela Politécnica Superior
Despacho - Módulo: B-417 Edificio B – 4ª Planta
Teléfono: +34 91 497 2253
Correo electrónico: Eduardo.perez@uam.es
Página web: http://www.eps.uam.es/esp/personal/ficha.php?empid=54
Horario de atención al alumnado: Petición de cita por correo electrónico.

1.11. Objetivos del curso

El estudiante obtendrá conocimientos prácticos sobre los métodos, prácticas, lenguajes y herramientas necesarios para el desarrollo en grupo de un proyecto software de tamaño medio, usando el paradigma de Orientación a Objetos. Estos

conocimientos se obtendrán de manera práctica mediante la realización de un proyecto en grupo.

Las **competencias** que se pretenden adquirir con esta asignatura son:

Comunes:
C1. Capacidad para diseñar, desarrollar, seleccionar y evaluar aplicaciones y sistemas informáticos, asegurando su fiabilidad, seguridad y calidad, conforme a principios éticos y a la legislación y normativa vigente.
C2. Capacidad para planificar, concebir, desplegar y dirigir proyectos, servicios y sistemas informáticos en todos los ámbitos, liderando su puesta en marcha y su mejora continua y valorando su impacto económico y social.
C3. Capacidad para comprender la importancia de la negociación, los hábitos de trabajo efectivos, el liderazgo y las habilidades de comunicación en todos los entornos de desarrollo de software.
C4. Capacidad para elaborar el pliego de condiciones técnicas de una instalación informática que cumpla los estándares y normativas vigentes.
C5. Conocimiento, administración y mantenimiento de sistemas, servicios y aplicaciones informáticas.
C8. Capacidad para analizar, diseñar, construir y mantener aplicaciones de forma robusta, segura y eficiente, eligiendo el paradigma y los lenguajes de programación más adecuados.
C16. Conocimiento y aplicación de los principios, metodologías y ciclos de vida de la ingeniería de software.
C17. Capacidad para diseñar y evaluar interfaces persona computador que garanticen la accesibilidad y usabilidad a los sistemas, servicios y aplicaciones informáticas.

De tecnología específica
IS1. Capacidad para desarrollar, mantener y evaluar servicios y sistemas software que satisfagan todos los requisitos del usuario y se comporten de forma fiable y eficiente, sean asequibles de desarrollar y mantener y cumplan normas de calidad, aplicando las teorías, principios, métodos y prácticas de la Ingeniería del Software.

Los objetivos que se pretenden alcanzar con esta asignatura son:

OBJETIVOS GENERALES	
G1	Analizar, Diseñar, Implementar y Probar programas usando tecnologías de Orientación a Objetos, que resulten en aplicaciones de alta calidad y mantenibles.
G2	Aplicar buenas prácticas, métodos, notaciones y herramientas de Ingeniería del Software en el desarrollo de aplicaciones en grupo.

OBJETIVOS ESPECIFICOS POR TEMA	
TEMA 1.- Requisitos	
1.1.	Capturar requisitos para una aplicación de tamaño medio.
1.2.	Representar y Analizar requisitos usando notaciones orientadas al flujo y orientadas a los escenarios.

TEMA 2.- Diseño	
2.1.	Diseñar la arquitectura de alto nivel de una aplicación software.
2.2.	Diseñar la estructura y comportamiento de una aplicación, en el paradigma orientado a objetos.
TEMA 3.- Implementación y Pruebas de Unidad	
3.1.	Implementar en equipo una aplicación de tamaño medio usando Java.
3.2.	Diseñar un conjunto de pruebas de unidad que garanticen un cierto nivel de confianza en la calidad del código desarrollado.
3.3.	Utilizar buenas prácticas de Ingeniería del Software, así como herramientas de pruebas tipo JUnit.
TEMA 4.- Pruebas	
4.1.	Realizar pruebas de integración, sistema y aceptación, que garanticen la calidad de la aplicación final.

1.12. Contenidos del programa

Programa Sintético

UNIDAD 1. Requisitos
UNIDAD 2. Diseño
UNIDAD 3. Implementación y Pruebas de Unidad
UNIDAD 4. Pruebas

Programa Detallado

1. **Requisitos**
   - 1.1. Captura.
   - 1.2. Notaciones.
     - 1.2.1. Orientadas al Flujo.
     - 1.2.2. Orientadas a los escenarios.
     - 1.2.3. Maquetas.
2. **Diseño**
   - 2.1. Arquitectura.
   - 2.2. Detallado.
3. **Implementación y Pruebas de Unidad.**
   - 3.1. Técnicas de programación en Java.
   - 3.2. Pruebas de Unidad. JUnit.
   - 3.3. Incrementos y pruebas de regresión.
4. **Pruebas**
   - 4.1. Pruebas de Integración.
   - 4.2. Pruebas de Sistema.
   - 4.3. Pruebas de Aceptación.

1.13. Referencias de consulta

Bibliografía:

Nota: Esta asignatura no sigue ningún libro en concreto. La lectura recomendada se lista por orden de afinidad al contenido del programa.

Unidad 1:

1. Software engineering a practitioner's approach, 7ªed. Roger Pressman. McGraw Hill Higher Education, 2010. También disponible en castellano.
2. Software engineering, 9ª ed. Addison Wesley. Ian Sommerville. INF/681.3.06/SOM. También disponible en castellano.
3. Software requirements styles and techniques. Lauesen, Soren. Addison-Wesley, 2002.
4. Ingeniería de software clásica y orientada a objetos, Sexta Edición. Stephen Schach. McGraw-Hill.
5. El lenguaje unificado de modelado manual de referencia. Rumbaugh, James. Pearson Addison Wesley. 2007.
6. Patrones de diseño elementos de software orientado a objetos reutilizable. Gamma, E., Helm, R., Johnson, R., Vlissides, J. Addison-Wesley, 2003.
7. Software Architecture in Practice (2nd Edition). Bass, Clements, Kazman. Addison-Wesley Professional, 2003.
8. Software Architecture: Foundations, Theory, and Practice. R. N. Taylor, N. Medvidovic, E. M. Dashofy, E. M. Dashofy. Wiley, 2010.
9. Designing the User Interface Strategies for effective human-computer interaction. Shneiderman, Ben. Pearson Education, 2005.

Unidad 3:

10. Core Java 2 Vol. 1 Fundamentos, Horstmann, Cay S. Prentice Hall, 2006. También disponible en castellano.
11. Core Java 2 Vol. 2 Características avanzadas, Horstmann, Cay S. Prentice Hall, 2006. También disponible en castellano.
12. Pruebas de software y JUnit: un análisis en profundidad y ejemplos prácticos. Bolaños, Sierra, Alarcón. Prentice-Hall, 2008.
13. Unit Testing in Java: How Tests Drive the Code. Link. Morgan Kaufmann; 1 edition, 2003.
14. Test Driven: TDD and Acceptance TDD for Java Developers. Koskela. Manning Publications, 2007.

Unidad 4:

15. Test Driven: TDD and Acceptance TDD for Java Developers. Koskela. Manning Publications, 2007.
16. Pruebas de software y JUnit: un análisis en profundidad y ejemplos prácticos. Bolaños, Sierra, Alarcón. Prentice-Hall, 2008.
17. Software engineering a practitioner's approach, 7ªed. Roger Pressman. McGraw Hill Higher Education, 2010. También disponible en castellano.

Nota: no se recomienda a los estudiantes comprar ningún libro hasta no haber comparado su contenido con el programa y revisado previamente en la biblioteca.

Material electrónico de trabajo: los documentos electrónicos de trabajo (recomendaciones sobre la elaboración de documentación, realización de diagramas, recomendaciones sobre legibilidad en el código) se publican en la sección de PADS en la plataforma Moodle (https://moodle.uam.es).

2. Métodos docentes

Con el objetivo de que los estudiantes adquieran conocimiento práctico para el desarrollo de un proyecto en equipo, la asignatura contendrá un conjunto de sesiones prácticas presenciales complementadas con trabajo regulado no presencial. Las sesiones presenciales se dividen en dos partes:

A) Explicación de las técnicas, métodos, notaciones y herramientas necesarias para la realización de una cierta fase del desarrollo.
B) Puesta en práctica de las técnicas explicadas en el contexto del proyecto software a desarrollar.

La metodología utilizada en el desarrollo de la actividad docente incluye los siguientes tipos de actividades:

***Prácticas:**

Actividad del profesor:

Explicación de las técnicas, métodos, notaciones y herramientas necesarias para la realización de una cierta fase del desarrollo.
Asignar una práctica/proyecto a cada grupo de trabajo y explicar la práctica asignada a cada grupo de trabajo al comienzo de la sesión de prácticas. Supervisar el trabajo de los grupos de trabajo en el laboratorio. Suministrar el guión de prácticas a completar en el laboratorio.
Los medios utilizados son los entornos de desarrollo y los ordenadores del propio laboratorio para el modelado, la ejecución, pruebas y análisis de los programas realizados.

Actividad del estudiante:

Actividad presencial: Dependiendo de la fase del desarrollo de la sesión en cuestión, los estudiantes deberán: i) comprender las explicaciones sobre las técnicas a emplear, ii) trabajar en equipo para aplicar dichas técnicas al desarrollo del proyecto, iii) redactar informes sobre los resultados obtenidos. En algunas sesiones, se requerirá la ejecución con el profesor presente, quien hará las preguntas oportunas a cada miembro del grupo para calificar de forma individual la práctica.
Actividad no presencial: Reuniones en equipo para finalizar el trabajo correspondiente. Redacción del informe de la práctica.

***Tutorías en aula:**

Actividad del profesor:

Tutorización a toda la clase o en grupos de alumnos reducidos (8-10) con el objetivo de resolver dudas comunes planteadas por los alumnos a nivel individual o en grupo, surgidas a partir del proyecto.

Actividad del estudiante:

Actividad presencial: Planteamiento de dudas individuales o en grupo y enfoque de posibles soluciones a las tareas planteadas.

Actividad no presencial: Estudio de las tareas marcadas y debate de las soluciones planteadas en el seno del grupo.

***Lecturas obligatorias y estudio personal**:

Actividad del estudiante:

Actividad no presencial: Aprendizaje autónomo académicamente dirigido por el profesor a través de las tareas publicadas en la página de docencia en red.

3. Tiempo de trabajo del estudiante

		Nº de horas	Porcentaje
Presencial	Clases presenciales	42 h (28%)	51 h (34%)
	Tutorías programadas a lo largo del semestre	6 h (4%)	
	Realización prueba final	3h (2%)	
No presencial	Trabajo independiente del estudiante	49 h (32.7%)	99 h (66%)
	Realización de actividades prácticas	50h (33.3%)	
Carga total de horas de trabajo: 25 horas x 6 ECTS		**150 h**	

4. Métodos de evaluación y porcentaje en la calificación final

- El proyecto se evaluará en 4 hitos, en los que el estudiante deberá entregar los entregables requeridos.
- La asignatura también incluye la realización de una prueba final, relacionada con las técnicas utilizadas en el desarrollo del proyecto.
- La nota final de la asignatura se obtiene por medio de la ecuación:

  Calificación Proyecto: 0.15*P1 + 0.15*P2 + 0.60*P3 + 0.10*P4

  Calificación Final en evaluación **continua**: 0.80*Proyecto + 0.20*Prueba final

  Calificación Final en evaluación **no continua**: 0.70*Proyecto + 0.30*Prueba final

- Dependiendo de las características del proyecto, es posible partir la entrega P3 en dos, cuyo peso total sería en cualquier caso del 60%.
- En la realización del proyecto no está permitido la utilización de herramientas para la generación automática de diagramas de clases UML ni para la generación automática de interfaces gráficas con el usuario.
- Para aprobar la asignatura es obligatorio obtener una nota mayor o igual a 5 puntos en cada uno de los entregables de la fase de codificación (P3), en la calificación del proyecto, así como en la prueba final.
- En la calificación del proyecto se tendrá en cuenta tanto la calidad del trabajo como su completitud. En particular, para aprobar es necesario implementar todos los requisitos de la aplicación, así como seguir un diseño que cumpla los principios de la orientación a objetos.
- En caso de no asistir al menos al 85% de las sesiones presenciales, el estudiante pasará a estar en **evaluación no continua**, en cuyo caso la prueba final individual tendrá una ponderación superior en el cálculo de la calificación final (ver fórmula anterior), sin que ello les exima de realizar todas y cada una de las entregas parciales del proyecto en las mismas fechas establecidas (comunes a los estudiantes en evaluación continua).
- La nota final en la convocatoria extraordinaria se calcula de la misma manera que en la convocatoria ordinaria. El proyecto en convocatoria extraordinaria consistirá en una extensión del proyecto realizado durante el curso. La nota de la prueba final y del proyecto se conservan (convalida) sólo para la convocatoria extraordinaria del mismo curso académico, pero no para el curso siguiente.
- El número mínimo de entregas para recibir una calificación numérica es 2. Por debajo de este número el estudiante recibirá la calificación "No evaluado". Aunque el estudiante no se presente a la prueba final, siempre que haya presentado este mínimo de entregables, recibirá una calificación numérica.

ATENCIÓN: Cualquier copia descubierta que se haya realizado a lo largo del curso, tanto en cualquiera de las actividades de teoría desarrolladas, como en el proyecto, serán penalizadas con rigurosidad, siguiendo la normativa vigente.

5. Cronograma

Semana	Contenido	Horas presenciales	Horas no presenciales (Trabajo autónomo del estudiante)
1	- Presentación de la asignatura, descripción del programa, normativa y los métodos de evaluación. - Presentación del proyecto a realizar. - **Unidad 1. Requisitos**	3	5 Realización del proyecto, fase de requisitos.
2	- **Unidad 1 Requisitos**	3	7 Realización del proyecto, fase de requisitos.
2	**Tutoría 1 sobre requisitos**	1	
3	- **Unidad 1 Requisitos** - **Unidad 2 Diseño**	3	7 Entrega material fase requisitos. Realización del proyecto, fase de diseño.
4	- **Unidad 2 Diseño**	3	6 Realización del proyecto, fase de diseño
5	- **Unidad 2 Diseño**	3	7 Realización del proyecto, fase de diseño
5	**Tutoría 2 sobre diseño**	2	
6	- **Unidad 3 Implementación y Pruebas de Unidad**	3	7 Entrega material fase diseño. Realización del proyecto, fase de diseño
7	- **Unidad 3 Implementación y Pruebas de Unidad**	3	6 Realización del proyecto, fase de implementación.
8	- **Unidad 3 Implementación y Pruebas de Unidad**	3	6 Realización del proyecto, fase de implementación
9	- **Unidad 3 Implementación y Pruebas de Unidad**	3	6 Realización del proyecto, fase de implementación
9	**Tutoría 3 sobre implementación y pruebas de Unidad**	1	
10	- **Unidad 3 Implementación y Pruebas de Unidad**	3	6 Realización del proyecto, fase de implementación
11	- **Unidad 3 Implementación y Pruebas de Unidad**	3	6 Realización del proyecto, fase de implementación

Semana	Contenido	Horas presenciales	Horas no presenciales (Trabajo autónomo del estudiante)
12	- **Unidad 3 Implementación y Pruebas de Unidad**	3	7 Realización del proyecto, fase de implementación
13	- **Unidad 4 Pruebas**	3	7 Entrega material fase implementación. Realización del proyecto, fase pruebas.
14	- **Unidad 4 Pruebas**	3	7 Entrega material fase pruebas.
	Tutoría 4, preparación para la prueba final	2	
	Prueba Final	3	10h