Guía docente de la asignatura

Asignatura	DESARROLLO DE APLICACIONES DISTRIBUIDAS		
Materia	INGENIERÍA DE REDES, SISTEMAS Y SERVICIOS TELEMÁTICOS		
Módulo	MATERIAS ESPECÍFICAS DE TECNOLOGÍAS DE TELECOMUNICACIÓN		
Titulación	GRADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN		
Plan	460	**Código**	45020
Periodo de impartición	1er CUATRIMESTRE	**Tipo/Carácter**	OBLIGATORIA
Nivel/Ciclo	GRADO	**Curso**	3º
Créditos ECTS	6 ECTS		
Lengua en que se imparte	CASTELLANO		
Profesor/es responsable/s	Jaime Gómez Gil		
Datos de contacto (E-mail, teléfono…)	TELÉFONO: 983 185556 E-MAIL: jgomez@tel.uva.es		
Horario de tutorías	Ver Tutorías del grado de Tecnologías en http://www.uva.es/export/sites/uva/2.docencia/2.01.grados/2.01.02.ofertaformativagrados/2.01.02.01.alfabetica/Grado-en-Ingenieria-de-Tecnologias-de-Telecomunicacion/		
Departamento	TEORÍA DE LA SEÑAL Y COMUNICACIONES E INGENIERÍA TELEMÁTICA		

1. Situación / Sentido de la Asignatura

1.1 Contextualización

La aparición y generalización de las redes de ordenadores ha dado lugar a la aparición de dos nuevos tipos de aplicaciones: las aplicaciones en red y las aplicaciones distribuidas. Estas aplicaciones se caracterizan por estar constituidas por varias partes que se pueden ejecutar en diferentes ordenadores. La coordinación entre estas partes se lleva a cabo mediante el intercambio de información a través de la red.

El desarrollo de una aplicación en red supone un esfuerzo superior al de una aplicación centralizada, ya que el desarrollador debe considerar todas las tareas de bajo nivel necesarias para el envío y recepción de información a través de la red y la gestión de los errores que puedan surgir durante esta comunicación. El desarrollo de las aplicaciones en red se basa en el uso de una API (interfaz de programación de aplicaciones) de acceso a los servicios de transporte de datos proporcionados por el sistema operativo. Esta API permite la solicitud de operaciones de bajo nivel relacionadas con el uso de la red (establecimiento y liberación de conexiones, envío y recepción de paquetes, etc.)

Para simplificar el desarrollo de las aplicaciones en red, de forma que la complejidad sea similar a la del desarrollo de una aplicación centralizada, surge el concepto de *middleware* o software intermediario. El *middleware* es, básicamente, una capa de software situada entre el sistema operativo y la aplicación a desarrollar, y que permite a la aplicación usar funciones de alto nivel (ejecución de una función en un ordenador remoto) sin necesidad de usar explícitamente las funciones de bajo nivel necesarias para la interacción con la red de comunicación. La aparición del *middleware* ha hecho surgir un nuevo tipo de aplicaciones, las aplicaciones distribuidas, que básicamente se diferencian de las aplicaciones en red en que el desarrollador no necesita incluir código para acceder a las funciones de bajo nivel relacionadas con la red (las operaciones sobre la red necesarias para acceder a un recurso remoto se realizan implícitamente mediante el software intermediario). En este concepto de *middleware* se basan las tecnologías para el desarrollo de aplicaciones distribuidas utilizadas mayoritariamente en la actualidad, como pueden ser *enterprise java beans, .Net o servicios web.*

La situación actual con redes ubicuas y de relativamente altas prestaciones ha dado lugar a un gran número de aplicaciones que necesitan usar una red de comunicaciones para proporcionar las funciones para las que han sido diseñadas. Por este motivo, resulta necesario en la actualidad formar profesionales que comprendan los principales problemas inherentes al desarrollo de aplicaciones en red y distribuidas y conozcan las principales tecnologías disponibles para desarrollar este tipo de aplicaciones de forma eficiente.

En esta asignatura se analizarán las principales diferencias entre aplicaciones centralizadas, en red y distribuidas y se explicará e ilustrará con ejemplos el proceso de desarrollo de aplicaciones en red y distribuidas siguiendo distintas tecnologías ampliamente usadas en la actualidad y que están basadas en los distintos tipos de *middleware* previamente presentados.

1.2 Relación con otras materias

Esta asignatura se apoya en la asignatura "Programación" de la materia "Informática" del "Bloque de Materias Instrumentales" que se imparte en el primer cuatrimestre del primer curso y en la asignatura de “Ingeniería de Sistemas Software” de la materia “Fundamentos de Sistemas Software” del “Bloque de Materias Básicas de Telecomunicaciones” que se imparte en el segundo cuatrimestre del segundo curso. En dichas asignaturas se proporcionarán los conceptos básicos de programación y desarrollo de sistemas software que facilitarán al alumno la comprensión de otros mostrados en la presente asignatura.

Además, también se utilizarán en esta asignatura conceptos relativos a la capa de transporte y red de la arquitectura de protocolos TCP/IP que han sido introducidos en la asignatura “Arquitectura de Redes, Sistemas y Servicios” de la materia “Fundamentos de Protocolos, Redes y Servicios Telemáticos” del “Bloque de Materias Básicas de Telecomunicaciones”, y que se imparte en el primer cuatrimestre del segundo curso.

1.3 Prerrequisitos

No existen condiciones previas excluyentes para cursar esta asignatura, aunque sí recomendaciones lógicas que el alumno debería tener en cuenta. En concreto, es recomendable haber cursado con anterioridad las asignaturas de “Programación”, “Ingeniería de Sistemas Software”, y “Arquitectura de Redes, Sistemas y Servicios”.

2. Competencias

2.1 Generales

- GBE3. Capacidad para resolver problemas con iniciativa, creatividad y razonamiento crítico.
- GC1. Capacidad de organización, planificación y gestión del tiempo.
- GC2. Capacidad para comunicar, tanto por escrito como de forma oral, conocimientos, procedimientos, resultados e ideas relacionadas con las telecomunicaciones y la electrónica.
- GC3. Trabajar en cualquier contexto, individual o en grupo, de aprendizaje o profesional, local o internacional, desde el respeto a los derechos fundamentales, de igualdad de sexo, raza o religión y los principios de accesibilidad universal, así como la cultura de paz.

2.2 Específicas

- T7 Conocimiento y utilización de los fundamentos de la programación en redes, sistemas y servicios de telecomunicación
- T13. Capacidad de diferenciar los conceptos de redes de acceso y transporte, redes de conmutación de circuitos y de paquetes, redes fijas y móviles, así como los sistemas y aplicaciones de red distribuidos, servicios de voz, datos, audio, vídeo y servicios interactivos y multimedia.
- TEL6. Capacidad de diseñar arquitecturas de redes y servicios telemáticos.
- TEL7. Capacidad de programación de servicios y aplicaciones telemáticas, en red y distribuidas.

3. Objetivos

Al finalizar la asignatura el alumno deberá ser capaz de:

- Describir los objetivos que se persiguen en el desarrollo de aplicaciones distribuidas.
- Comprender la problemática específica asociada al desarrollo de aplicaciones en distribuidas.
- Comprender los conceptos relacionados con el *middleware* como arquitectura básica para el desarrollo de aplicaciones distribuidas.
- Conocer las técnicas básicas sobre las que se basan las aplicaciones distribuidas.
- Diseñar, desarrollar y desplegar aplicaciones distribuidas utilizando una API de acceso a los servicios de transporte de datos.
- Diseñar, desarrollar y desplegar aplicaciones distribuidas utilizando tecnologías basadas en *middleware*.

4. Tabla de dedicación del estudiante a la asignatura

ACTIVIDADES PRESENCIALES	HORAS	ACTIVIDADES NO PRESENCIALES	HORAS
Clases teórico-prácticas (T/M)	30	Estudio y trabajo autónomo individual	45
Clases prácticas de aula (A)	0	Estudio y trabajo autónomo grupal	45
Laboratorios (L)	30		
Prácticas externas, clínicas o de campo	0		
Seminarios (S)	0		
Tutorías grupales (TG)			
Evaluación (fuera del periodo oficial de exámenes)	0		
Total presencial	**60**	**Total no presencial**	**90**

5. Bloques temáticos

Bloque 1: Desarrollo de aplicaciones distribuidas

Carga de trabajo en créditos ECTS: 6

a. Contextualización y justificación

Véase el apartado 1.1.

b. Objetivos de aprendizaje

Véase el apartado 3.

c. Contenidos

El temario a impartir en las **clases teórico-prácticas** es el siguiente:

TEMA 1: Introducción a la computación distribuida

- Definiciones
- Historia de la computación distribuida
- Diferentes formas de computación
- Virtudes y limitaciones
- Conceptos básicos de sistemas distribuidos

TEMA 2: Nombres en sistemas distribuidos

- Nombres, identificadores y direcciones
- Nombres planos
- Nombres estructurados
- Tablas HASH distribuidas

TEMA 3: Sincronización en sistemas distribuidos

- Relojes físicos
- Sincronización de relojes físicos
- Relojes lógicos
- Relojes Vectoriales

TEMA 4: Tolerancia a fallos en sistemas distribuidos

- Disponibilidad, confiabilidad, seguridad, mantenimiento y otros conceptos básicos
- Modelos de fallas
- Disfrazado de fallas por redundancia
- Enmascaramiento de fallas y replicación

Acuerdos en sistemas defectuosos

El problema del acuerdo bizantino

TEMA 5: IPC-Comunicación entre procesos

Arquetipo para la comunicación entre procesos

Sincronización de eventos

Temporizadores e hilos de ejecución

Interbloqueos y temporizadores

Representación y codificación de datos

Protocolos basados en texto y protocolos de solicitud-respuesta

Diagramas de eventos y de secuencia

Comunicación orientada y no orientada a conexión

Evolución de los paradigmas de comunicación entre procesos

TEMA 6: Paradigmas de computación distribuida

Paradigmas y abstracción

Paradigmas para aplicaciones distribuidas

Comparativa

TEMA 7: El API de *sockets*

Antecedentes

La metáfora de *sockets* en IPC

El API de *sockets* en modo *stream*

Sockets con operaciones de E/S no bloqueantes

El API de *sockets* seguros

TEMA 8: El paradigma cliente-servidor

Antecedentes

Cuestiones sobre el paradigma cliente-servidor

Ingeniería de software de un servicio de red

Servidor iterativo y servidor concurrente

Servidores con estado

TEMA 9: Comunicación de grupo

Unidifusión frente a multidifusión

Una API de multidifusión arquetípica

Multidifusión sin conexión frente orientada a conexión

El API de multidifusión básica de Java

TEMA 10: Objetos distribuidos

Paso de mensajes frente a objetos distribuidos

Una arquitectura típica de objetos distribuidos

Sistemas de objetos distribuidos

Llamadas a procedimientos remotos (RPC) e Invocaciones a Métodos Remotos (RMI)

La arquitectura de Java RMI

El API de Java RMI

Una aplicación RMI de ejemplo

Comparación entre RMI y el API de *Sockets*

Se realizarán prácticas en el **laboratorio** que tendrán las siguientes características:

- Cada práctica durará una jornada de laboratorio.
- Cada práctica estará formada por diferentes pruebas, en cada una de las cuales el alumno tiene que realizar una aplicación JAVA concreta.
- Las pruebas normalmente serán modificaciones simples de pruebas anteriores, para añadir poco a poco funcionalidades a las aplicaciones que se desarrollan.

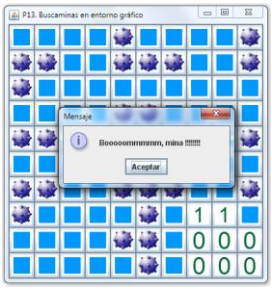
En las prácticas de laboratorio el alumno programará, entre otras, las siguientes aplicaciones Java:

- Aplicación hundir la flota con interfaz en modo texto. Las siguientes líneas ilustran el funcionamiento de esta aplicación.

```
Introduzca el número de fila (0-8): 6
Introduzca el número de columna (0-9): 8
El disparo cae en un barco que aún no se ha hundido
Los disparos realizados son:
          0 0 0 0 0 0 0 0 0 0
          0 0 0 0 0 0 0 0 0 0
          0 0 0 0 0 0 0 0 0 0
          0 0 0 0 0 0 0 0 0 0
          0 0 0 0 0 0 0 0 0 0
          0 0 0 0 0 0 0 0 0 0
          0 0 0 0 0 0 0 0 2 0
          0 0 0 0 0 0 0 0 0 0
          0 0 0 0 0 0 0 0 0 0
```

- Comunicador en modo texto que permite a dos alumnos chatear entre sí desde dos ordenadores diferentes conectados a Internet. Realizado empleando Sockets Java. La siguiente imagen ilustra el funcionamiento de esta aplicación.

- Aplicación buscaminas en entorno gráfico, en versión cliente-servidor, empleando el API RMI de Java. La siguiente imagen ilustra el funcionamiento de esta aplicación.



En las jornadas de laboratorio el alumno realizará las aplicaciones Java, usando, entre otros:

- La clase *Thread,* que permite el desarrollo de aplicaciones con varios flujos de ejecución.
- El API de *Sockets* Java, que permite la comunicación entre procesos de una forma simple.
- El API de Java RMI, que permite la comunicación entre procesos de una forma más elaborada.
- El API de *Swing*, que permite la programación del entorno gráfico de aplicaciones Java.
- El Inferface *MouseListener,* que permite el manejo de eventos del ratón en Java.
- El modelo vista-controlador, que permitirá realizar aplicaciones en entorno gráfico con una capa asociada únicamente a la interacción con el usuario.

d. Métodos docentes

- Clase magistral participativa.
- Taller de prácticas guiadas en el laboratorio.

e. Plan de trabajo

Véase el Anexo I.

f. Evaluación

La evaluación de la adquisición de competencias se basará en:

- Una prueba escrita al final del cuatrimestre en la que se evaluarán los conocimientos adquiridos tanto en las clases teórico-prácticas como en el laboratorio.
- Evaluaciones del trabajo del alumno en el laboratorio, que se realizará mediante la valoración de:
  - Los informes escritos del trabajo realizado por el alumno
  - Las respuestas de los alumnos a las preguntas planteadas por el profesor en el laboratorio.
  - La actitud y proactividad del alumno en el laboratorio.

g. Bibliografía básica

- A.S.Tananbaum & M.V. Steen. *Distributed Systems: Princicples and Paradigms*, 2nd ed. Pearson-Prentice Hall, 2007.
- M.l. Liu. *Distributed Computing, principles and applications*. Prentice-Hall 2004

- P. Deitel & H. Deitel. *Java how to program*, 9th ed. Prentice-Hall, 2012.

h. Bibliografía complementaria

- G. F. Coulouris, J. Dollimore & T. Kindberg. *Distributed systems: concepts and design*, 5th ed. Addison-Wesley, 2012.
- K. Sierra & B. Bates. *Head First Java*, 2nd ed. O'Reil, 2005.
- R. W. Stevens. *Unix Network Programming. Networking APIs: sockets and XTI*, 2a ed. Prentice-Hall, 1998.
- W. Emmerich. *Engineering distributed objects*. John Wiley & Sons, 2000.
- C. Szyperski. *Component software: beyond object oriented programming*. Addison Wesley, 1999.
- G. Alonso, F. Casati, H. Kuno & V. Machiraju. *Web Services: Concepts, Architectures and Applications*. Springer, 2004.
- L. Richardson, S. Ruby & D. H. Hansson. *RESTful web services*. O'Reilly, 2007.
- P. Deitel & H. Deite. *Java, how to program, ninth edition*. Prentice Hall. 2012.
- J. Pritchard. *COM and CORBA side by side : architectures, strategies, and implementations.* Addison-Wesley, 1999.
- G. Suresh Raj. *A Detailed Comparison of CORBA, DCOM and Java/RMI*. 2000. http://my.execpc.com/~gopalan/misc/compare.html
- Oracle: *Java Platform, Standard Edition 7 API Specification*. 2011. http://java.sun.com/javase/7/docs/api/
- Oracle: *Java Platform, Enterprise Edition (Java EE). Documentation home page.* http://java.sun.com/j2ee
- Oracle: *The java tutorials: RMI*. 2012. http://download.oracle.com/javase/tutorial/rmi/index.html
- Oracle: *Dynamic code downloading using RMI*. 2003. http://download.oracle.com/javase/1.4.2/docs/guide/rmi/codebase.html
- Oracle: *Java SE documentation: Java Remote Method Invocation (Java RMI)*. 2011. http://download.oracle.com/javase/7/docs/technotes/guides/rmi/index.html
- Oracle: *Java SE documentation: Java RMI over IIOP*. 2011. http://download.oracle.com/javase/7/docs/technotes/guides/rmi-iiop/index.html
- *The java tutorials: Java Naming and Directory Interface*. 2012. http://download.oracle.com/javase/tutorial/jndi/index.html
- Oracle: *Java SE Documentation. Java IDL (CORBA).* 2011. http://java.sun.com/javase/6/docs/technotes/guides/idl/
- *Microsoft Component Object Model Technologies.* 2012. http://www.microsoft.com/com/default.mspx
- *E. Roman, S. Ambler, & T. Jewell, Mastering Enterprise JavaBeans, 2.a ed. Wiley, 2002.*
- *D. Sevilla, Tutorial de CCM. http://ditec.um.es/~dsevilla/ccm/#CCMtutorial*
- *D. C. Schmidt, Página web de Douglas C. Schmidt sobre CCM.* http://www.cs.wustl.edu/~schmidt/corba.html
- G. Suresh Raj, *Component Engineering Cornucopia*.. http://my.execpc.com/~gopalan/
- Object Management Group: Tutorial sobre CCM. http://www.omg.org/news/meetings/workshops/RT_2003_Manual/Tutorials/T3_CCM_Wang-Rodrigues.pdf
- Microsoft: *Microsoft .NET*. http://www.microsoft.com/net

- Codeguru: *The .NET Architecture*. http://www.codeguru.com/csharp/sample_chapter/article.php/c8245
- W3C: *Web of services. 2012.* http://www.w3.org/standards/webofservices/
- W3C: *Web Services Activity*. 2011. http://www.w3.org/2002/ws

i. Recursos necesarios

- Documentación de apoyo
- Entorno de trabajo en la plataforma Moodle ubicada en el Campus Virtual de la Universidad de Valladolid u otra plataforma virtual alternativa.
- Laboratorio de prácticas, con al menos un ordenador para cada dos alumnos, para las sesiones de laboratorio.

6. Temporalización (por bloques temáticos)

BLOQUE TEMÁTICO	CARGA ECTS	PERIODO PREVISTO DE DESARROLLO
Bloque 1: Desarrollo de aplicaciones distribuidas	6 ECTS	Semanas 1 a 15

7. Sistema de calificaciones – Tabla resumen

INSTRUMENTO/PROCEDIMIENTO	PESO EN LA NOTA FINAL	OBSERVACIONES
INSTRUMENTO 1: Examen final escrito sobre los conocimientos asociados tanto a las clases teórico-prácticas, **como a los contenidos sobre programación Java asociados al laboratorio**. Denominaremos a la nota de este instrumento ***E***.	60%	Si el alumno no realiza el INSTRUMENTO 1, entonces, la nota del alumno en la convocatoria será la de "no presentado". Esto es válido para cualquier convocatoria. Será necesario que $E \geq 5$ para aprobar la asignatura. En caso de que $E<5$, la nota final de la asignatura será E, independientemente de la nota L. Esto es válido para cualquier convocatoria. La nota final de la asignatura cuando $E \geq 5$ será ***máximo{E, E·0.6+ L·0.4}***. Esto es válido para cualquier convocatoria. Aunque no es imprescindible, es muy recomendable que el alumno asista al laboratorio, pues en el INSTRUMENTO 1 se evaluarán tanto los contenidos impartidos en las clases teórico-prácticas, como los contenidos asociados a las prácticas en lenguaje Java que el alumno realiza en el laboratorio.
INSTRUMENTO 2: Evaluación del trabajo de alumno en el laboratorio. Denominaremos a la nota de este instrumento ***L***.	40%	

No se mantiene ninguna nota de un curso académico para el siguiente.

. Consideraciones finales

El Anexo I mencionado en la guía, donde se describe la planificación detallada, se entregará al comienzo de la asignatura.