

	MODELO DE PROGRAMACIÓN ANUAL	MD7101	
		VERSIÓN 0	Pág. 1 de 51

Programación anual del módulo profesional

Programación

Ricardo Pérez López

Curso 2019/2020

Departamento de Informática y Comunicaciones
Jefe de Departamento: Ricardo Pérez López

Índice

1. Información general	3
2. Objetivos generales	3
3. Resultados de aprendizaje y criterios de evaluación	4
4. Instrumentos y procedimientos de evaluación y calificación	8
4.1. Valoración general de los contenidos	9
4.2. Calificación	9
4.2.1. Calificaciones parciales	10
4.2.2. Calificación final	11
4.3. Competencias básicas	11
4.4. Medidas de recuperación	12
4.4.1. Recuperación del apartado TRA	12
4.4.2. Recuperación del apartado EXA	13
5. Contenidos y temporalización	13
5.1. Cuadro resumen	14
5.2. Esquema detalle	14
5.3. Correspondencia con resultados de aprendizaje y criterios de evaluación	40
6. Orientaciones pedagógicas	43
7. Orientaciones metodológicas	44
8. Recursos	46
8.1. Hardware	46
8.2. Software	46
8.3. Online	46
8.4. Bibliografía	46
8.4.1. Principal	46
8.4.2. Complementaria	47
9. Atención a la diversidad	47
10. Temas transversales	47
11. Actuaciones para desarrollar la perspectiva de género	49
11.1. Actuaciones generales permanentes	49

	MODELO DE PROGRAMACIÓN ANUAL	MD7101	
		VERSIÓN 0	Pág. 3 de 51

1. Información general

- **Normativa de aplicación:** [1], [2]
- **Equivalencia en créditos ECTS:** 14.
- **Código:** 0485.
- **Duración total:** 256 horas (32 semanas)
- **Carga lectiva semanal:** 8 horas

2. Objetivos generales

1. La formación del módulo contribuye a alcanzar los **objetivos generales** de este ciclo formativo que se relacionan a continuación [1]:
 - e) Interpretar el diseño lógico, verificando los parámetros establecidos para gestionar bases de datos.
 - j) Emplear herramientas y lenguajes específicos, siguiendo las especificaciones, para desarrollar componentes multimedia.
 - q) Programar y realizar actividades para gestionar el mantenimiento de los recursos informáticos.
2. La formación del módulo contribuye a alcanzar las **competencias profesionales, personales y sociales** de este título que se relacionan a continuación [1]:
 - a) Configurar y explotar sistemas informáticos, adaptando la configuración lógica del sistema según las necesidades de uso y los criterios establecidos.
 - e) Desarrollar aplicaciones Web con acceso a bases de datos utilizando lenguajes, objetos de acceso y herramientas de mapeo adecuados a las especificaciones.
 - f) Integrar contenidos en la lógica de una aplicación Web, desarrollando componentes de acceso a datos adecuados a las especificaciones.
 - i) Integrar componentes multimedia en el interface de una aplicación Web, realizando el análisis de interactividad, accesibilidad y usabilidad de la aplicación.
 - j) Desarrollar e integrar componentes software en el entorno del servidor Web, empleando herramientas y lenguajes específicos, para cumplir las especificaciones de la aplicación.
 - v) Realizar la gestión básica para la creación y funcionamiento de una pequeña empresa y tener iniciativa en su actividad profesional con sentido de la responsabilidad social.

	MODELO DE PROGRAMACIÓN ANUAL	MD7101	
		VERSIÓN 0	Pág. 4 de 51

3. Resultados de aprendizaje y criterios de evaluación

Los resultados de aprendizaje del módulo y sus criterios de evaluación asociados son los que se describen a continuación [1]:

[RA1] Reconoce la estructura de un programa informático, identificando y relacionando los elementos propios del lenguaje de programación utilizado.

Criterios de evaluación:

- CE1.a)* Se han identificado los bloques que componen la estructura de un programa informático.
- CE1.b)* Se han creado proyectos de desarrollo de aplicaciones.
- CE1.c)* Se han utilizado entornos integrados de desarrollo.
- CE1.d)* Se han identificado los distintos tipos de variables y la utilidad específica de cada uno.
- CE1.e)* Se ha modificado el código de un programa para crear y utilizar variables.
- CE1.f)* Se han creado y utilizado constantes y literales.
- CE1.g)* Se han clasificado, reconocido y utilizado en expresiones los operadores del lenguaje.
- CE1.h)* Se ha comprobado el funcionamiento de las conversiones de tipo explícitas e implícitas.
- CE1.i)* Se han introducido comentarios en el código.

[RA2] Escribe y prueba programas sencillos, reconociendo y aplicando los fundamentos de la programación orientada a objetos.

Criterios de evaluación:

- CE2.a)* Se han identificado los fundamentos de la programación orientada a objetos.
- CE2.b)* Se han escrito programas simples.
- CE2.c)* Se han instanciado objetos a partir de clases predefinidas.
- CE2.d)* Se han utilizado métodos y propiedades de los objetos.
- CE2.e)* Se han escrito llamadas a métodos estáticos.
- CE2.f)* Se han utilizado parámetros en la llamada a métodos.
- CE2.g)* Se han incorporado y utilizado librerías de objetos.
- CE2.h)* Se han utilizado constructores.

	MODELO DE PROGRAMACIÓN ANUAL	MD7101	
		VERSIÓN 0	Pág. 5 de 51

CE2.i) Se ha utilizado el entorno integrado de desarrollo en la creación y compilación de programas simples.

[RA3] Escribe y depura código, analizando y utilizando las estructuras de control del lenguaje.

Criterios de evaluación:

CE3.a) Se ha escrito y probado código que haga uso de estructuras de selección.

CE3.b) Se han utilizado estructuras de repetición.

CE3.c) Se han reconocido las posibilidades de las sentencias de salto.

CE3.d) Se ha escrito código utilizando control de excepciones.

CE3.e) Se han creado programas ejecutables utilizando diferentes estructuras de control.

CE3.f) Se han probado y depurado los programas.

CE3.g) Se ha comentado y documentado el código.

[RA4] Desarrolla programas organizados en clases analizando y aplicando los principios de la programación orientada a objetos.

Criterios de evaluación:

CE4.a) Se ha reconocido la sintaxis, estructura y componentes típicos de una clase.

CE4.b) Se han definido clases.

CE4.c) Se han definido propiedades y métodos.

CE4.d) Se han creado constructores.

CE4.e) Se han desarrollado programas que instancien y utilicen objetos de las clases creadas anteriormente.

CE4.f) Se han utilizado mecanismos para controlar la visibilidad de las clases y de sus miembros.

CE4.g) Se han definido y utilizado clases heredadas.

CE4.h) Se han creado y utilizado métodos estáticos.

CE4.i) Se han definido y utilizado interfaces.

CE4.j) Se han creado y utilizado conjuntos y librerías de clases.

[RA5] Realiza operaciones de entrada y salida de información, utilizando procedimientos específicos del lenguaje y librerías de clases.

Criterios de evaluación:

CE5.a) Se ha utilizado la consola para realizar operaciones de entrada y salida de información.

	MODELO DE PROGRAMACIÓN ANUAL		MD7101
			VERSIÓN 0 Pág. 6 de 51

CE5.b) Se han aplicado formatos en la visualización de la información.

CE5.c) Se han reconocido las posibilidades de entrada / salida del lenguaje y las librerías asociadas.

CE5.d) Se han utilizado ficheros para almacenar y recuperar información.

CE5.e) Se han creado programas que utilicen diversos métodos de acceso al contenido de los ficheros.

CE5.f) Se han utilizado las herramientas del entorno de desarrollo para crear interfaces gráficos de usuario simples.

CE5.g) Se han programado controladores de eventos.

CE5.h) Se han escrito programas que utilicen interfaces gráficos para la entrada y salida de información.

[RA6] Escribe programas que manipulen información seleccionando y utilizando tipos avanzados de datos.

Criterios de evaluación:

CE6.a) Se han escrito programas que utilicen arrays

CE6.b) Se han reconocido las librerías de clases relacionadas con tipos de datos avanzados.

CE6.c) Se han utilizado listas para almacenar y procesar información.

CE6.d) Se han utilizado iteradores para recorrer los elementos de las listas.

CE6.e) Se han reconocido las características y ventajas de cada una de la colecciones de datos disponibles.

CE6.f) Se han creado clases y métodos genéricos.

CE6.g) Se han utilizado expresiones regulares en la búsqueda de patrones en cadenas de texto.

CE6.h) Se han identificado las clases relacionadas con el tratamiento de documentos XML.

CE6.i) Se han realizado programas que realicen manipulaciones sobre documentos XML.

[RA7] Desarrolla programas aplicando características avanzadas de los lenguajes orientados a objetos y del entorno de programación.

Criterios de evaluación:

CE7.a) Se han identificado los conceptos de herencia, superclase y subclase.

	MODELO DE PROGRAMACIÓN ANUAL	MD7101	
		VERSIÓN 0	Pág. 7 de 51

CE7.b) Se han utilizado modificadores para bloquear y forzar la herencia de clases y métodos.

CE7.c) Se ha reconocido la incidencia de los constructores en la herencia.

CE7.d) Se han creado clases heredadas que sobrescriban la implementación de métodos de la superclase.

CE7.e) Se han diseñado y aplicado jerarquías de clases.

CE7.f) Se han probado y depurado las jerarquías de clases.

CE7.g) Se han realizado programas que implementen y utilicen jerarquías de clases.

CE7.h) Se ha comentado y documentado el código.

[RA8] Utiliza bases de datos orientadas a objetos, analizando sus características y aplicando técnicas para mantener la persistencia de la información.

Criterios de evaluación:

CE8.a) Se han identificado las características de las bases de datos orientadas a objetos.

CE8.b) Se ha analizado su aplicación en el desarrollo de aplicaciones mediante lenguajes orientados a objetos.

CE8.c) Se han instalado sistemas gestores de bases de datos orientados a objetos.

CE8.d) Se han clasificado y analizado los distintos métodos soportados por los sistemas gestores para la gestión de la información almacenada.

CE8.e) Se han creado bases de datos y las estructuras necesarias para el almacenamiento de objetos.

CE8.f) Se han programado aplicaciones que almacenen objetos en las bases de datos creadas.

CE8.g) Se han realizado programas para recuperar, actualizar y eliminar objetos de las bases de datos.

CE8.h) Se han realizado programas para almacenar y gestionar tipos de datos estructurados, compuestos y relacionados.

[RA9] Gestiona información almacenada en bases de datos relacionales manteniendo la integridad y consistencia de los datos.

Criterios de evaluación:

CE9.a) Se han identificado las características y métodos de acceso a sistemas gestores de bases de datos relacionales.

CE9.b) Se han programado conexiones con bases de datos.

CE9.c) Se ha escrito código para almacenar información en bases de datos.

CE9.d) Se han creado programas para recuperar y mostrar información almacenada en bases de datos.

CE9.e) Se han efectuado borrados y modificaciones sobre la información almacenada.

CE9.f) Se han creado aplicaciones que ejecuten consultas sobre bases de datos.

CE9.g) Se han creado aplicaciones para posibilitar la gestión de información presente en bases de datos relacionales.

4. Instrumentos y procedimientos de evaluación y calificación

La evaluación tendrá como finalidad determinar el nivel de competencia de los alumnos y la consecución de los objetivos. Se desarrollará de forma continua, y atenderá a los siguientes aspectos:

- Aprendizaje autónomo, viendo la capacidad del alumno para interiorizar, gestionar y participar en los procesos de aprendizaje propios.
- Comprensión del lenguaje común.
- Adquisición de conceptos básicos del módulo profesional que permiten al alumno incluirlos como un elemento más de su realidad profesional.
- Participación y trabajo en grupo, viendo la capacidad que tiene este de escuchar y debatir las diferentes soluciones de un problema.
- Nivel de abstracción alcanzado.

Para que el seguimiento de dicha evaluación sea factible, el alumno deberá asistir a clase con regularidad y participar activamente en la misma, de forma que una falta sistemática supondrá la **pérdida de la evaluación continua** y sólo tendrá derecho a un examen final. Asimismo, se requiere que el alumno acceda al menos diariamente a la **plataforma Ágora** y que revise diariamente su correo en el dominio @iesdonana.org para informarse puntualmente de las novedades en el módulo. Es responsabilidad del alumno informarse **activamente** de las novedades del módulo.

4.1. Valoración general de los contenidos

Los contenidos se ponderarán en base a los siguientes porcentajes:

Trabajos, actividades y ejercicios (casa, clase, grupo) (TRA)	30 %
Pruebas evaluativas (EXA)	70 %

Si, por algún motivo, no se pudiera evaluar uno de los dos apartados anteriores (**TRA** o **EXA**), el otro apartado restante soportaría el 100 % de la carga evaluativa, de forma que la calificación final resultaría únicamente de dicho apartado.

4.2. Calificación

Se llevarán a cabo trabajos, actividades y/o ejercicios (apartado **TRA**) versados sobre los contenidos trabajados en una unidad didáctica o bloque de unidades didácticas conceptualmente relacionadas. (Esto significa, en consecuencia, que no es obligatoria la realización de trabajos, actividades y/o ejercicios en cada unidad didáctica, sino que a tales efectos se pueden agrupar varias unidades didácticas.)

Dentro de este grupo podrá incluirse alguna prueba (tipo test o de respuestas cortas) a responder de forma individual sobre conocimientos teóricos de determinados aspectos básicos asociados a unidades (o conjunto de unidades) didácticas concretas.

Asimismo, se realizará un examen al final de cada evaluación parcial (apartado **EXA**), coincidiendo aproximadamente con el final del trimestre correspondiente. Debido al carácter de evaluación continua del módulo, así como del hecho de que cada contenido trabajado se asienta sobre los anteriores, es posible que el alumno requiera, para la evaluación positiva de un trimestre, el conocimiento necesario de contenidos de trimestres anteriores, independientemente de que dichos conocimientos hayan sido evaluados positiva o negativamente en trimestres anteriores.

Cada examen, a su vez, constará de dos partes bien diferenciadas:

Parte teórica: Una prueba que versará sobre conocimientos esenciales con preguntas tipo test, de respuestas cortas o similar (apartado **TEO**).

Parte práctica: Una prueba práctica con problemas y ejercicios donde el alumno deberá aplicar lo aprendido escribiendo código real (apartado **PRA**).

La nota del examen se calculará como una media ponderada de ambas partes (como se recoge en el siguiente epígrafe) y para superar el examen en su conjunto será necesario obtener una calificación mínima de 3 puntos en cada parte del mismo.

	MODELO DE PROGRAMACIÓN ANUAL	MD7101	
		VERSIÓN 0	Pág. 10 de 51

4.2.1. Calificaciones parciales

La **calificación de cada evaluación parcial** (primer, segundo y tercer trimestres por separado) se calculará de la siguiente forma:

Algoritmo 1 (Cálculo de la calificación parcial)

```

if (mínimo(TEO, PRA)  $\geq$  3) {
    EXA = TEO * 0.3 + PRA * 0.7;
} else {
    EXA = mínimo(TEO, PRA);
}

if (mínimo(TRA, EXA)  $\geq$  4) {
    NOTA = TRA * 0.3 + EXA * 0.7;
} else {
    NOTA = mínimo(TRA, EXA);
}

```

Donde:

TRA: Media aritmética de las calificaciones de los trabajos realizados en ese trimestre, valorados del 0 al 10.

TEO: Calificación de la parte teórica del examen correspondiente a ese trimestre, valorada del 0 al 10.

PRA: Calificación de la parte práctica del examen correspondiente a ese trimestre, valorada del 0 al 10.

EXA: Calificación del examen correspondiente a ese trimestre, valorada del 0 al 10, calculada como la media ponderada de **TEO** y **PRA**.

La evaluación parcial se considera aprobada si **NOTA** \geq 4,5.

Si durante el trimestre en cuestión no se realizaran trabajos, actividades y/o ejercicios (apartado **TRA**) dignos de calificación, entonces los porcentajes se redistribuirán de forma que la calificación de la evaluación parcial correspondiente coincidirá con la nota del examen (apartado **EXA**), por lo que quedará simplemente de la siguiente forma:

$$\mathbf{NOTA = EXA}$$

Asimismo, si durante el trimestre en cuestión no se realizaran exámenes (apartado **EXA**), entonces los porcentajes se redistribuirán de forma que la calificación de la evaluación parcial

correspondiente coincidirá con la nota de los trabajos, actividades y/o ejercicios (apartado **TRA**), por lo que quedará simplemente de la siguiente forma:

$$\text{NOTA} = \text{TRA}$$

Las **faltas de ortografía** en los exámenes serán **penalizadas** según acuerdo del Departamento, de la siguiente forma:

Número de faltas	Evaluación 1	Evaluación 2
< 5	–0, 25 puntos	–0, 50 puntos
≥ 5	–0, 50 puntos	–1, 00 puntos

En ningún caso este criterio podrá ser motivo de que el alumno no supere la prueba escrita.

4.2.2. Calificación final

La calificación final del módulo se calculará de la siguiente forma:

Algoritmo 2 (Cálculo de la calificación final)

```

if (mínimo(EV1, EV2, EV3) ≥ 4) {
    NOTA = EV1 * 0.5 + EV2 * 0.5 + EV3 * 0.5;
} else {
    NOTA = mínimo(EV1, EV2, EV3);
}
  
```

Donde:

EV₁: Calificación de la primera evaluación.

EV₂: Calificación de la segunda evaluación.

EV₃: Calificación de la tercera evaluación.

El módulo se considera aprobado si **NOTA** ≥ 4,5.

4.3. Competencias básicas

La superación del módulo irá supeditado a la verificación de que el alumno ha alcanzado las **competencias básicas** del mismo, que son las que se enumeran a continuación:

CE1.e) Se han identificado y caracterizado los principales lenguajes y tecnologías relacionados con la programación Web en entorno servidor.

	MODELO DE PROGRAMACIÓN ANUAL	MD7101	
		VERSIÓN 0	Pág. 12 de 51

- CE2.d) Se ha reconocido la sintaxis del lenguaje de programación que se ha de utilizar.
- CE2.g) Se han utilizado los distintos tipos de variables y operadores disponibles en el lenguaje.
- CE2.h) Se han identificado los ámbitos de utilización de las variables.
- CE3.a) Se han utilizado mecanismos de decisión en la creación de bloques de sentencias.
- CE3.b) Se han utilizado bucles y se ha verificado su funcionamiento.
- CE3.c) Se han utilizado «arrays» para almacenar y recuperar conjuntos de datos.
- CE3.d) Se han creado y utilizado funciones.
- CE3.e) Se han utilizado formularios Web para interactuar con el usuario del navegador Web.
- CE3.f) Se han empleado métodos para recuperar la información introducida en el formulario.
- CE3.g) Se han añadido comentarios al código.
- CE4.b) Se han utilizado sesiones para mantener el estado de las aplicaciones Web.
- CE4.e) Se han escrito aplicaciones que integren mecanismos de autenticación de usuarios.
- CE5.f) Se han escrito aplicaciones Web con mantenimiento de estado y separación de la lógica de negocio.
- CE5.g) Se han aplicado los principios de la programación orientada a objetos.
- CE6.f) Se han creado aplicaciones Web que permitan la actualización y la eliminación de información disponible en una base de datos.

Para verificar que se han alcanzado las competencias básicas anteriores, se tendrán en cuenta las calificaciones obtenidas en cada trabajo, actividad y ejercicio de clase, así como las obtenidas en cada uno de los ejercicios de los que se compongan los exámenes realizados.

4.4. Medidas de recuperación

4.4.1. Recuperación del apartado TRA

A lo largo del curso se abrirán nuevas ventanas temporales para que el alumnado entregue las correcciones necesarias en aquellas actividades que estén pendientes de evaluación positiva. La evaluación de tales correcciones podrá requerir la defensa de las mismas en una entrevista individual con el alumno para garantizar la autoría y la adquisición adecuada de las competencias correspondientes.

	MODELO DE PROGRAMACIÓN ANUAL	MD7101	
		VERSIÓN 0	Pág. 13 de 51

4.4.2. Recuperación del apartado EXA

Al final de cada evaluación, o a comienzos de la evaluación siguiente, se llevará a cabo una prueba de recuperación para aquellos alumnos que tengan algún contenido pendiente en esa evaluación. Al final del curso se hará una prueba final de recuperación para aquellos alumnos que tengan pendientes contenidos de alguna evaluación, debiendo presentarse únicamente a aquellas partes que tengan pendientes.

5. Contenidos y temporalización

Cada unidad didáctica tiene una duración temporal de **una semana**, que podrá ampliarse o reducirse en función de las circunstancias cuando el profesor lo estime conveniente (por ejemplo, para la realización de actividades, ejercicios y prácticas en clase).

Los contenidos marcados con la etiqueta *#opcional* son contenidos complementarios que sólo se impartirán si hay tiempo suficiente para ello y nunca a costa de otros contenidos no opcionales. También podrán ser usados como elementos a desarrollar para el alumnado con altas capacidades o que manifieste un ritmo de aprendizaje superior al del resto del grupo/clase.

Todas las fechas se muestran en formato ISO 8601 (*año-mes-día*).

5.1. Cuadro resumen

Evaluación	Unidad didáctica	Inicio estimado
Primera	1. Sistemas de control de versiones I	2019-09-16
	2. Sistemas de control de versiones II	2019-09-23
	3. Introducción a la tecnología web	2019-09-30
	4. Conceptos básicos de PHP I	2019-10-07
	5. Conceptos básicos de PHP II	2019-10-14
	6. Desarrollo de aplicaciones con PHP I	2019-10-21
	7. Persistencia de datos con PHP	2019-10-28
	8. Desarrollo de aplicaciones con PHP II	2019-11-04
	9. Programación avanzada en PHP	2019-11-11
	10. Interoperabilidad	2019-11-18
	11. Estructura de una aplicación Yii 2	2019-12-02
Segunda	12. Gestión de peticiones en Yii 2	2020-01-07
	13. Acceso a bases de datos en Yii 2	2020-01-13
	14. Creación y validación de formularios en Yii 2	2020-01-20
	15. Visualización de datos en Yii 2	2020-01-27
	16. Seguridad y cacheado en Yii 2	2020-02-03
	17. Características adicionales de Yii 2	2020-02-10
	18. Calidad	2020-02-17
	19. Computación en la nube	2020-02-24
	20. Servicios web con REST en Yii 2 <i>#opcional</i>	
	21. Contenedores <i>#opcional</i>	

5.2. Esquema detalle

Cuando un resultado de aprendizaje no lleva asociado ningún criterio de evaluación, significa que dicho resultado de aprendizaje se trabaja en la unidad didáctica pero no es el elemento fundamental de evaluación.

1 **Sistemas de control de versiones I** *#ev1* (est: 2019-09-16) _____

1.1. Preparación del entorno de desarrollo

1.1.1. Instalación automatizada

- Acciones previas
 - Instalar git
 - Crear cuenta en GitHub
 - Solicitar el Student Developer Pack

- Usar <https://github.com/ricpelo/conf> y seguir las instrucciones del README.md

1.1.2. Terminal

- Zsh
- Oh My Zsh
- less

1.1.3. Navegador

1.1.4. Editores de texto

- Vim y less
- Atom
 - Instalación
 - Configuración
 - Paquetes
- Alternativa: PhpStorm

1.1.5. DokuWiki *#opcional*

- Elaboración de documentación
- La wiki como sistema de control de versiones
- La wiki como herramienta colaborativa

1.2. Primeros pasos

1.2.1. config

1.2.2. git-config.sh

1.2.3. init

1.2.4. add

1.2.5. commit

- Con la opción -m
- Sin la opción -m

1.2.6. checkout (descartar cambios)

1.2.7. reset

1.2.8. .gitignore

1.3. Estado

1.3.1. status

1.3.2. log

1.3.3. Alias lg

1.3.4. show

1.3.5. diff

- git diff
- git diff --staged

- `git diff <commit>`
- `git diff inicial..final`

1.3.6. Referencias

- HEAD y master
- 237ab45^
- 237ab45~1

1.4. La máquina del tiempo

- 1.4.1. checkout (mover el HEAD)
- 1.4.2. revert
- 1.4.3. reset
- 1.4.4. tag
- 1.4.5. --amend

1.5. Borrar y mover

- 1.5.1. rm
- 1.5.2. mv

1.6. Git y los directorios

1.7. Resultados de aprendizaje y criterios de evaluación asociados

- 1.7.1. RA1: CE1.e, CE1.g
- 1.7.2. RA4: CE4.g

2 Sistemas de control de versiones II #ev1 (est: 2019-09-23)

2.1. Ramas locales

- 2.1.1. branch
- 2.1.2. merge
 - Estrategia *fast-forward*
 - Estrategia *recursive* o del padre múltiple
 - --no-ff

2.1.3. Resolución de conflictos

2.1.4. rebase

2.2. Ramas remotas

2.2.1. Orígenes remotos

- Directorios locales
- Servidores remotos con repositorios compartidos
- `remote [add|show] origin`

2.2.2. Flujo de trabajo básico

- push
 - Ramas de seguimiento (*tracking branch*)

- clone
 - ¿Qué significa origin/HEAD?

- fetch
- pull

2.2.3. Eliminar ramas remotas

2.2.4. Etiquetas remotas

- `git push origin mi_etiqueta`
- `git push --tags`
- `git push --delete origin mi_etiqueta`

2.3. GitHub

2.3.1. El flujo de trabajo de GitHub

2.3.2. Pull requests

- Comentarios generales y comentarios en línea
- Revisiones de cambios
 - Crear y solicitar revisiones
- Arreglar una PR
- Cerrar una PR
 - `git remote prune origin`

2.3.3. Issues

2.3.4. Releases

2.3.5. Forks

2.3.6. GitHub Education

- GitHub Classroom

2.4. Meta

2.4.1. Objetivos de la unidad

2.4.2. *Resultados de aprendizaje y criterios de evaluación asociados*

- RA1: CE1.e, CE1.g
- RA4: CE4.g

3 Introducción a la tecnología web [#ev1](#) (est: 2019-09-30) _____

3.1. Introducción al desarrollo web

3.1.1. Conceptos básicos

- Navegadores y servidores web
- Agentes de usuario
- Web estática vs. dinámica
- Estructura vs. contenido

- Arquitectura multinivel

3.1.2. Ejemplos de aplicaciones web

- Redes sociales: Facebook, Twitter...
- Comercio electrónico: Amazon, eBay...
- Administración electrónica...
- Portales
- ERP, CRM

3.1.3. Tecnologías de desarrollo de aplicaciones web

- .NET
- Java
- Ruby/Rails
- Python/Django
- PHP
- El Kung-Fu de la Programación
 - Odoo
 - PrestaShop
 - Drupal
 - WordPress

3.2. Arquitectura cliente/servidor

3.3. HTML 5 básico (recordatorio de primer curso)

3.4. Protocolo HTTP

3.4.1. URIs

- URL encoding

3.4.2. Peticiones (*HTTP requests*) y respuestas (*HTTP responses*)

3.4.3. Métodos: GET, POST

3.4.4. Versiones

- HTTP/1.0
- HTTP/1.1

3.4.5. Cabeceras HTTP

3.4.6. Códigos de estado

3.4.7. Experimentos

- telnet (a un servidor)
- netcat (desde un navegador)
- `curl -i -XPOST "http://..." | pygmentize -l http`
- http
- Google Chrome Developer Tools

3.4.8. Envío de datos al servidor

- Mediante GET
- Mediante POST
- Formularios HTML

3.4.9. Cookies

3.5. Apache básico *#opcional*

3.5.1. Instalación

3.5.2. Configuración básica

3.5.3. Sitios virtuales

3.6. Resultados de aprendizaje y criterios de evaluación asociados

3.6.1. RA1: CE1.a, CE1.b, CE1.c, CE1.d, CE1.e, CE1.g

4 Conceptos básicos de PHP I *#ev1* (est: 2019-10-07) _____

4.1. Introducción a PHP

4.1.1. Página web de PHP

4.1.2. Instalación de PHP

4.1.3. Documentación y búsqueda de información

4.2. Sintaxis básica

4.2.1. Datos e instrucciones

4.2.2. Sentencias y comandos

- Comando echo

4.2.3. Expresiones, operadores y funciones

4.3. Funcionamiento del intérprete

4.3.1. Ejecución

- Por lotes
- Interactiva
 - php -a
 - PsySH

4.3.2. Etiquetas <?php y ?>

4.3.3. Modo dual de operación

4.4. Variables

4.4.1. Conceptos básicos

4.4.2. Destrucción de variables

4.4.3. Operadores de asignación por valor y por referencia

4.4.4. Variables predefinidas

4.5. Tipos básicos de datos

4.5.1. Lógicos (bool)

- Operadores lógicos

4.5.2. Numéricos

- Enteros (int)
- Números en coma flotante (float)
- Operadores
 - Operadores aritméticos
 - Operadores de incremento/decremento

4.5.3. Cadenas (string)

- Operadores de cadenas
 - Concatenación
 - Acceso y modificación por caracteres
 - Operador de incremento *#opcional*
- Funciones de manejo de cadenas
- Extensión *mbstring*

4.5.4. Nulo (null)

4.6. Manipulación de datos

4.6.1. Precedencia de operadores

4.6.2. Operadores de asignación compuesta

4.6.3. Comprobaciones

- De tipos
 - `gettype()`
 - `is_*()`
- De valores
 - `is_numeric()`
 - `ctype_*()`

4.6.4. Conversiones de tipos

- Conversión explícita (forzado o *casting*) vs. automática
- Conversión a bool
- Conversión a int
- Conversión a float
- Conversión de string a número
- Conversión a string
- Funciones de obtención de valores
 - `intval()`

	MODELO DE PROGRAMACIÓN ANUAL	MD7101	
		VERSIÓN 0	Pág. 21 de 51

- floatval()
- strval()
- boolval()

- Funciones de formateo numérico
 - number_format()
 - money_format()
 - setlocale()

4.6.5. Comparaciones

- Operadores de comparación
- == vs. ===
- Ternario (?:)
- Fusión de null (??)
- Reglas de comparación de tipos

4.7. Constantes

- 4.7.1. define() y const
- 4.7.2. Constantes predefinidas
- 4.7.3. defined()

4.8. Resultados de aprendizaje y criterios de evaluación asociados

- 4.8.1. RA2: CE2.c, CE2.d, CE2.e, CE2.f, CE2.g, CE2.h
- 4.8.2. RA3
- 4.8.3. RA4: CE4.g

5 Conceptos básicos de PHP II *#ev1* (est: 2019-10-14) _____

5.1. Flujo de control

5.1.1. Estructuras de control

- Secuencia
- Selección
- Iteración
- Sintaxis alternativa

5.1.2. Inclusión de scripts

- include, require
- include_once, require_once

5.2. Funciones predefinidas destacadas

- 5.2.1. isset()
- 5.2.2. empty()
- 5.2.3. var_dump()

5.3. Arrays

5.3.1. Operadores para arrays

- Acceso, modificación y agregación

5.3.2. Funciones de manejo de arrays

- Ordenación de arrays
- `print_r()`
- `'+'` vs. `array_merge()`
- `isset()` vs. `array_key_exists()`

5.3.3. `foreach`

5.3.4. Conversión a array

5.3.5. *Ejemplo*: `$argv` en CLI

5.4. Funciones definidas por el usuario

5.4.1. Argumentos

- Paso de argumentos por valor y por referencia
- Argumentos por defecto

5.4.2. Ámbito de variables

- Ámbito simple al archivo
- Variables locales
- Uso de `global`
- Variables superglobales

5.4.3. Declaraciones de tipos

- Declaraciones de tipo de argumento
- Declaraciones de tipo de devolución
- Tipos *nullable* (?) y `void`
- Tipificación estricta

5.5. Comentarios y documentación del código

5.6. *Resultados de aprendizaje y criterios de evaluación asociados*

5.6.1. RA2: CE2.d, CE2.e, CE2.g

5.6.2. RA3: CE3.a, CE3.b, CE3.c, CE3.d, CE3.g

5.6.3. RA4: CE4.g

6 Desarrollo de aplicaciones con PHP I #ev1 (est: 2019-10-21) _____

6.1. SAPIs

6.1.1. CLI: Uso en línea de comandos

- `$argc` y `$argv`
- Flujos de entrada/salida

6.1.2. Apache

- Integración de PHP con Apache
- PHP como lenguaje embebido
- Etiqueta <?=
- Servidor web interno

6.1.3. CGI: PHP-FPM (FastCGI Process Manager)

6.1.4. Configuración básica con php.ini

- `error_reporting = E_ALL`
- `display_errors = On`
- `display_startup_errors = On`
- `date.timezone = 'UTC'`

6.1.5. Módulos de extensión

6.2. Manejo de datos de entrada: `$_GET` y `$_POST`

6.3. Funciones auxiliares interesantes

6.3.1. `extract()`

6.3.2. `compact()`

6.4. Aspectos básicos de la orientación a objetos

6.4.1. Objetos

- `new`
- `instanceof`

6.4.2. Referencias

- Asignación por referencia (`=&`)

6.4.3. Clonación de objetos

6.4.4. Comparación de objetos

6.4.5. Propiedades

- Predeterminadas
- Dinámicas

6.4.6. Métodos

6.4.7. Constantes

- Operador de resolución de ámbito (`::`)

6.4.8. *Ejemplo*: manejo de fechas, horas, instantes e intervalos

6.5. Excepciones

6.5.1. Manejo de errores clásico en PHP

6.5.2. Errores vs. excepciones

6.5.3. La clase `Exception`

6.5.4. La clase `Error`

	MODELO DE PROGRAMACIÓN ANUAL		MD7101
			VERSIÓN 0 Pág. 24 de 51

6.5.5. La clase `ErrorException`

6.5.6. Estructura de control `try ... catch`

6.6. Depuración

6.6.1. `var_dump()`, `print_r()`, `die()`

6.6.2. `PsySH`

6.6.3. `Xdebug` *#opcional*

- Módulo `Xdebug`
- Aplicación `Xdebug` para Chrome
- Extensión `Xdebug Helper` para Chrome
- Paquete `php-debug` para Atom

6.7. Resultados de aprendizaje y criterios de evaluación asociados

6.7.1. RA1: *CE1.a, CE1.b, CE1.d, CE1.f*

6.7.2. RA2: *CE2.a, CE2.b, CE2.c, CE2.d, CE2.e, CE2.f, CE2.g, CE2.h*

6.7.3. RA3: *CE3.a, CE3.b, CE3.c, CE3.d, CE3.e, CE3.f, CE3.g*

6.7.4. RA4: *CE4.g*

6.7.5. RA5: *CE5.d, CE5.g, CE5.h*

7 Persistencia de datos con PHP *#ev1* (est: 2019-10-28) _____

7.1. PDO (PHP Data Objects)

7.1.1. Clase `PDO`

- `__construct(string $dsn [, string $username [, string $password [, array $options]]])`
- `PDOStatement query(string $statement)`
- `int exec(string $statement)`
- `PDOStatement prepare(string $statement [, array $driver_options = array()])`

7.1.2. Clase `PDOStatement`

- `mixed fetch([int $fetch_style])`
- `mixed fetchAll([int $fetch_style])`
- `mixed fetchColumn([int $column_number = 0])`
- `bool execute ([array $input_parameters])`
- `int rowCount(void)`

7.1.3. Correspondencias de tipos entre SQL y PHP

7.1.4. Transacciones

- `$pdo->beginTransaction();`
- `$pdo->commit();`
- `$pdo->rollBack();`

7.2. Cookies

7.2.1. setcookie()

7.2.2. Ejemplos de uso

7.3. Sesiones

7.3.1. Iniciar una sesión

- session_start()

7.3.2. Usar una sesión

- \$_SESSION
- Ejemplos de uso

7.3.3. Terminar una sesión

- session_destroy()
- session_name()
- session_id()
- session_get_cookie_params()

7.4. Seguridad y persistencia

7.4.1. Contraseñas

- <https://www.md5online.org/>
- <https://www.sha1online.org/>
- password_hash()
- password_verify()

7.4.2. Inyección de código SQL

7.4.3. Cross-Site Request Forgery (CSRF)

7.5. Resultados de aprendizaje y criterios de evaluación asociados

7.5.1. RA2

7.5.2. RA3

7.5.3. RA4: CE4.a, CE4.b, CE4.c, CE4.g

7.5.4. RA5: CE5.f, CE5.g, CE5.h

7.5.5. RA6: CE6.a, CE6.b, CE6.c, CE6.e, CE6.g

8 Desarrollo de aplicaciones con PHP II *#ev1* (est: 2019-11-04) _____

8.1. Programación de *CRUD* básico

8.1.1. Ejemplo de aplicación: *Muéveme*

8.1.2. Ejemplo de aplicación: *FilmAffinity*

8.2. Post/Redirect/Get

8.3. header()

8.3.1. output_buffering

	MODELO DE PROGRAMACIÓN ANUAL	MD7101	
		VERSIÓN 0	Pág. 26 de 51

8.4. Seguridad básica

8.4.1. Filtrar la entrada, escapar la salida

8.4.2. Cross-Site Scripting (XSS)

- No persistente
- Persistente
- Escapado de la salida
 - htmlspecialchars()
 - HTML Purifier

8.4.3. Filtrado de la entrada

- Cómo *NO* se debe hacer
- Extensión Filter
 - filter_input(), filter_has_var(), filter_var()
 - Filtros de validación y saneado
- Expresiones regulares (PCRE)

8.5. Resultados de aprendizaje y criterios de evaluación asociados

8.5.1. RA1: CE1.a

8.5.2. RA4: CE4.a, CE4.b, CE4.c, CE4.d, CE4.e, CE4.f, CE4.g

8.5.3. RA5: CE5.a, CE5.b, CE5.d, CE5.f, CE5.g, CE5.h

8.5.4. RA6: CE6.a, CE6.b, CE6.c, CE6.d, CE6.e, CE6.f, CE6.g, CE6.h

9 Programación avanzada en PHP #ev1 (est: 2019-11-11) _____

9.1. Diseño de aplicaciones orientadas a objetos

9.1.1. Clases

9.1.2. Propiedades

- Predeterminadas
- Dinámicas

9.1.3. Métodos

- Referencia \$this
- Constructores y destructores

9.1.4. Constantes

- self

9.1.5. Herencia

- parent
- Sobreescritura de métodos

9.1.6. Miembros estáticos

- Constantes

- Métodos estáticos
- Propiedades estáticas
- Enlace estático en tiempo de ejecución

9.1.7. Interfaces

9.1.8. Traits

9.1.9. La clase `std::class`

- Conversión de array a object.

9.2. Espacios de nombres

9.3. Funciones anónimas

9.3.1. Clausuras

9.4. Callables

9.4.1. `call_user_func()`

9.4.2. `array_map()` y `array_reduce()`

9.5. *Resultados de aprendizaje y criterios de evaluación asociados*

9.5.1. RA2

9.5.2. RA3

9.5.3. RA4: CE4.g

9.5.4. RA5: CE5.g, CE5.h

10 Interoperabilidad #ev1 (est: 2019-11-18) _____

10.1. Versionado semántico

10.2. Composer

10.2.1. Paquetes

10.2.2. Packagist

10.2.3. Dependencias

- `composer.json` y `composer.lock`

10.2.4. Versiones y restricciones

- Versión exacta
- Rango (`>`, `>=`, `<`, `<=`, `!=`, `,`, `,`, `|`)
- Guión (`-`)
- Asterisco (`*`)
- Tilde (`~`)
- Gorrito (`^`)
- Nombres de rama
 - `dev-master`
 - `5.1.x-dev`
- Estabilidad mínima

- Comprobador online de restricciones

10.2.5. Comandos básicos

- require
- install
- update

10.2.6. Entornos de desarrollo y producción

10.3. Autocarga de clases

10.3.1. spl_autoload_register()

10.3.2. PSR-4

10.3.3. Autoloader de Composer

10.4. Ejemplos

10.4.1. mpdf/mpdf

10.4.2. ramsey/uuid

10.4.3. doctrine/inflector

10.5. Recomendaciones PSR del PHP-FIG (Framework Interop Group)

10.5.1. PSR-1: Basic Coding Standard

10.5.2. PSR-2: Coding Style Guide

10.5.3. PSR-4: Autoloading Standard

10.5.4. PSR-5: PHPDoc Standard (borrador)

10.5.5. PSR-11: Extended Coding Style Guide (borrador)

10.5.6. PSR-19: PHPDoc tags (borrador)

10.6. Paquetes de Atom y herramientas externas *#opcional*

10.6.1. PHP_CodeSniffer

10.6.2. PHP-CS-Fixer

10.6.3. Yii2-Shell

10.7. Resultados de aprendizaje y criterios de evaluación asociados

10.7.1. RA1: CE1.e, CE1.g

10.7.2. RA4: CE4.g

10.7.3. RA5: CE5.g, CE5.h

10.7.4. RA9: CE9.a, CE9.b, CE9.e

11 Introducción a Yii 2 *#ev1* (est: 2019-11-25)

11.1. Frameworks, microframeworks y librerías

11.2. Patrón Modelo-Vista-Controlador (MVC)

11.2.1. Modelos

11.2.2. Vistas

11.2.3. Controladores

- 11.2.4. Rutas
- 11.3. Yii 2
 - 11.3.1. ¿Qué es Yii?
 - 11.3.2. ¿En qué es mejor Yii?
 - 11.3.3. ¿Cómo es Yii comparado con otros frameworks?
 - 11.3.4. Versiones de Yii
- 11.4. Instalación, requisitos y puesta en marcha
 - 11.4.1. Requisitos previos
 - 11.4.2. Instalación de Yii 2
 - Instalación mediante Composer
 - 11.4.3. Plantillas de proyecto
 - Plantilla básica vs. avanzada
 - Plantilla básica modificada
- 11.5. ¡Hola, mundo!
- 11.6. Formularios
- 11.7. Bases de datos
- 11.8. Generador de código Gii
- 11.9. Herramientas de desarrollo
 - 11.9.1. Barra de depuración
 - 11.9.2. Trazas de depuración
- 11.10. Estilo del código
- 11.11. *Resultados de aprendizaje y criterios de evaluación asociados*
 - 11.11.1. RA1: CE1.e, CE1.g
 - 11.11.2. RA4: CE4.g
 - 11.11.3. RA5: CE5.a, CE5.b, CE5.c, CE5.d, CE5.e, CE5.g, CE5.h
 - 11.11.4. RA6: CE6.a, CE6.b, CE6.c, CE6.d, CE6.e, CE6.f, CE6.g, CE6.h
 - 11.11.5. RA9: CE9.b, CE9.c, CE9.e, CE9.f, CE9.g

12 Estructura de una aplicación Yii 2 *#ev1* (est: 2019-12-02)

- 12.1. A pequeña escala
 - 12.1.1. Componentes
 - La clase yii\base\BaseObject
 - Propiedades
 - Configuraciones
 - Asignación masiva
 - Creación de nuevas instancias

	MODELO DE PROGRAMACIÓN ANUAL	MD7101	
		VERSIÓN 0	Pág. 30 de 51

- Normas de creación de componentes
- Diferencias entre `new` y `Yii::createObject()`
- La clase `yii\base\Component`
 - Eventos
 - De instancia
 - Eventos de instancia
 - Manejadores de eventos de instancia
 - De clase
 - Eventos de clase
 - Manejadores de eventos de clase
 - Comportamientos

12.1.2. Alias

12.1.3. Autoloading de clases

12.1.4. Localizador de servicios

12.1.5. Contenedor de inyección de dependencias

12.2. A gran escala

12.2.1. Introducción

12.2.2. Scripts de entrada

12.2.3. Aplicaciones

12.2.4. Componentes de aplicación

12.2.5. Controladores

- Acciones
- Filtros

12.2.6. Modelos

12.2.7. Vistas

- Widgets

12.2.8. Otros componentes

- Módulos
- Assets
- Extensiones

12.3. Resultados de aprendizaje y criterios de evaluación asociados

12.3.1. RA2

12.3.2. RA3

12.3.3. RA4: CE4.g

12.3.4. RA5: CE5.g, CE5.h

12.3.5. RA9: CE9.e, CE9.f, CE9.g

13 Gestión de peticiones en Yii 2 #ev2 (est: 2020-01-07)

- 13.1. Introducción
- 13.2. Arranque (bootstrapping)
- 13.3. Enrutado y creación de URLs
- 13.4. Peticiones
- 13.5. Respuestas
- 13.6. Sesiones y cookies
- 13.7. *Resultados de aprendizaje y criterios de evaluación asociados*
 - 13.7.1. RA4: CE4.g
 - 13.7.2. RA5: CE5.d, CE5.f, CE5.g, CE5.h

14 Acceso a bases de datos en Yii 2 #3d #ev2 (est: 2020-01-13)

- 14.1. DAO
 - 14.1.1. yii\db\Connection
 - 14.1.2. yii\db\Connection::createCommand()
 - 14.1.3. Consultas SQL
 - queryAll()
 - queryOne()
 - queryColumn()
 - queryScalar()
 - 14.1.4. Sentencias no SELECT
 - execute()
 - insert()
 - update()
 - delete()
- 14.2. Query Builder
 - 14.2.1. yii\db\Query
 - 14.2.2. Creación de consultas
 - select()
 - from()
 - Condiciones y filtrado de filas
 - where()
 - Formatos de condiciones
 - De cadena
 - De array
 - De operadores
 - andWhere()

- `orWhere()`
- `filterWhere()`
- `andFilterWhere()`
- `orFilterWhere()`
- `orderBy()`
- `groupBy()`
- Condiciones y filtrado de grupos
 - `having()`
 - `filterHaving()`
 - `andFilterHaving()`
 - `orFilterHaving()`
- `limit()`
- `offset()`
- Combinaciones
 - `join()`
 - `innerJoin()`
 - `leftJoin()`
 - `rightJoin()`
- `union()`

14.2.3. Recogida de resultados

- `all()`
- `one()`
- `column()`
- `scalar()`
- `exists()`
- `count()`
- Funciones de grupo
 - `sum()`
 - `average()`
 - `max()`
 - `min()`
- `indexBy()`

14.2.4. Consultas por lotes

- `batch()`
- `each()`

14.3. Active Record

14.3.1. `findOne()`

14.3.2. findAll()

14.3.3. save()

14.3.4. ActiveQuery

- find()

14.3.5. Atributos sucios

14.3.6. Relaciones

- Encadenamiento de relaciones

14.3.7. joinWith()

14.3.8. Atributos virtuales

- Siete técnicas
 - Calcular a mano cuando/donde haga falta
 - Usar vistas SQL
 - Sobreescibir el método find() del modelo para que se use siempre en lugar del heredado de ActiveRecord
 - Sobreescibir el método afterFind() para rellenar el atributo a mano cada vez que se hace un find()
 - Capturar el evento EVENT_AFTER_FIND del modelo
 - Usar una propiedad con getter y setter
 - Crear un método findEspecial() que se usará en lugar de find() cuando haga falta
 - La mejor opción, en la mayoría de los casos: combinar las dos anteriores
 - Ejemplo

14.4. *Resultados de aprendizaje y criterios de evaluación asociados*

14.4.1. RA2

14.4.2. RA3

14.4.3. RA4: CE4.g

14.4.4. RA5: CE5.f, CE5.g, CE5.h

14.4.5. RA6: CE6.a, CE6.b, CE6.c, CE6.d, CE6.e, CE6.f, CE6.g, CE6.h

14.4.6. RA9: CE9.e, CE9.f, CE9.g

15 Creación y validación de formularios en Yii 2 #ev2 (est: 2020-01-20) _____

15.1. Creación de formularios

15.1.1. ActiveForm

15.1.2. yii\helpers\Html

15.2. Validación de la entrada

15.2.1. Declaración de reglas

- Validadores principales

- skipOnEmpty
- skipOnError
- Personalizar mensajes de error
- Validación condicional
- Filtrado (saneado) de datos
- Manejo de entradas vacías

15.2.2. Validadores en línea

15.2.3. Validaciones en el cliente *#opcional*

15.3. Subida de archivos

15.4. *Resultados de aprendizaje y criterios de evaluación asociados*

15.4.1. RA2

15.4.2. RA3: CE3.e, CE3.f

15.4.3. RA4: CE4.g

15.4.4. RA5: CE5.b, CE5.d, CE5.h

15.4.5. RA6: CE6.f, CE6.h

15.4.6. RA8: CE8.a, CE8.b, CE8.c, CE8.d, CE8.e, CE8.f, CE8.g

15.4.7. RA9: CE9.e, CE9.f, CE9.g

16 Visualización de datos en Yii 2 *#ev2* (est: 2020-01-27)

16.1. Formateado de datos

16.1.1. yii\i18n\Formatter

16.2. Paginación

16.2.1. yii\data\Pagination

16.2.2. Entrada

- totalCount
- pageSize
- page

16.2.3. Salida

- limit
- offset

16.2.4. yii\widgets\LinkPager

16.3. Ordenación

16.3.1. yii\data\Sort

16.3.2. Entrada

- attributes
- sort

16.3.3. Salida

- orders

16.3.4. yii\data\Sort::link()

16.4. Proveedores de datos

16.4.1. Entrada

- pagination
- sort

16.4.2. Salida

- models
- count
- totalCount

16.4.3. ActiveDataProvider

- query

16.4.4. SqlDataProvider

- sql
- params
- totalCount

16.4.5. ArrayDataProvider

- allModels

16.5. Widgets de datos

16.5.1. DetailView

16.5.2. ListView

16.5.3. GridView

- dataProvider
- columns
 - DataColumn
 - SerialColumn
 - ActionColumn
- Ordenación de columnas
- Filtrado de datos
 - yii\grid\GridView::\$filterModel
- Relaciones
 - Ordenación con relaciones
 - Filtrado con relaciones
 - yii\db\ActiveRecord::getAttribute(\$name)

16.5.4. El problema de las fechas/horas/instantes

16.5.5. Otros widgets no oficiales

- Krajee Yii Extensions
 - yii2-datecontrol
 - yii2-number

16.6. Scripts de cliente

16.7. Temas *#opcional*

16.8. Resultados de aprendizaje y criterios de evaluación asociados

16.8.1. RA2

16.8.2. RA3

16.8.3. RA4: CE4.g

16.8.4. RA5: CE5.b, CE5.c, CE5.d, CE5.h

16.8.5. RA6: CE6.f, CE6.h

16.8.6. RA8: CE8.a, CE8.b, CE8.c, CE8.d, CE8.e, CE8.f, CE8.g

16.8.7. RA9: CE9.e, CE9.f, CE9.g

17 Seguridad y cacheado en Yii 2 *#ev2* (est: 2020-02-03)

17.1. Autenticación

17.1.1. Componente de aplicación user

17.1.2. Clase identidad e interfaz yii\web\IdentityInterface

17.1.3. Métodos de *login* y *logout*.

17.2. Contraseñas

17.3. Autorización

17.4. Niveles de caché

17.4.1. Cacheado de datos

17.4.2. Cacheado de fragmentos

17.4.3. Cacheado de páginas

17.4.4. Cacheado HTTP

17.5. Resultados de aprendizaje y criterios de evaluación asociados

17.5.1. RA2

17.5.2. RA3

17.5.3. RA4: CE4.d, CE4.e, CE4.g

17.5.4. RA5: CE5.e, CE5.g, CE5.h

17.5.5. RA6: CE6.h

17.5.6. RA8

17.5.7. RA9: CE9.e, CE9.f, CE9.g

18 Características adicionales de Yii 2 #ev2 (est: 2020-02-10)

18.1. AJAX y PJAX

18.1.1. Validaciones Ajax

18.1.2. PJAX #opcional

18.1.3. CORS #opcional

18.2. Correo electrónico

18.3. Aplicación de consola

18.4. Migraciones

18.5. Extensiones

18.6. Paquetes

18.7. Resultados de aprendizaje y criterios de evaluación asociados

18.7.1. RA2

18.7.2. RA3

18.7.3. RA4: CE4.g

18.7.4. RA5: CE5.d, CE5.g, CE5.h

18.7.5. RA6: CE6.h

18.7.6. RA8: CE8.a, CE8.b, CE8.c, CE8.d, CE8.e, CE8.f, CE8.g

18.7.7. RA9: CE9.e, CE9.f, CE9.g

19 Calidad #ev2 (est: 2020-02-17)

19.1. Pruebas

19.1.1. Tipos de pruebas

- Unitarias
- Funcionales
- De aceptación

19.1.2. Herramientas

- PHPUnit #opcional
- Codeception
 - Ejecutar pruebas
 - Crear pruebas en formato Cest
- Fixtures
 - ./yii fixture/generate <nombre>
 - ./yii fixture/load <nombre>

19.1.3. Integración continua: Travis CI

19.1.4. Cobertura de código #opcional

19.2. Depuración

19.2.1. var_dump() mejorado

	MODELO DE PROGRAMACIÓN ANUAL	MD7101	
		VERSIÓN 0	Pág. 38 de 51

- 19.2.2. Consola integrada
- 19.2.3. Barra de depuración
- 19.2.4. Depuración con PsySH *#opcional*
- 19.3. Documentación
 - 19.3.1. API documentation generator for Yii2
 - 19.3.2. GitHub Pages
- 19.4. Mantenimiento y calidad del código
 - 19.4.1. CodeSniffer
 - 19.4.2. CS_Fixer
 - 19.4.3. Code Climate
- 19.5. *Resultados de aprendizaje y criterios de evaluación asociados*
 - 19.5.1. RA3: CE3.g
 - 19.5.2. RA4: CE4.g
 - 19.5.3. RA5: CE5.g, CE5.h
 - 19.5.4. RA6: CE6.e, CE6.h
 - 19.5.5. RA9: CE9.e, CE9.g

20 **Computación en la nube #ev2** (est: 2020-02-24)

- 20.1. Entornos de ejecución
 - 20.1.1. Desarrollo
 - 20.1.2. Producción
 - 20.1.3. Pruebas
 - 20.1.4. Preproducción
- 20.2. Cloud computing vs hosting
- 20.3. Cloud computing vs VPS
- 20.4. Servicios por capas
 - 20.4.1. IaaS
 - 20.4.2. PaaS
 - 20.4.3. SaaS
- 20.5. 12 Factores
- 20.6. Heroku
 - 20.6.1. Heroku CLI
 - 20.6.2. Creación y despliegue de aplicaciones
 - 20.6.3. Heroku Postgres
 - 20.6.4. Variables de entorno
 - 20.6.5. Releases
- 20.7. Escalabilidad *#opcional*

20.8. Alta disponibilidad *#opcional*

20.9. *Resultados de aprendizaje y criterios de evaluación asociados*

20.9.1. RA1: CE1.a, CE1.c, CE1.d, CE1.e

20.9.2. RA5: CE5.h

20.9.3. RA6: CE6.h

20.9.4. RA9: CE9.e, CE9.f, CE9.g

21 Servicios web con REST en Yii 2 *#ev2 #opcional* (est: 2020-03-02) _____

21.1. Introducción

21.2. Recursos

21.3. Controladores

21.4. Enrutado

21.5. Formateo de la respuesta

21.6. Autenticación

21.7. Limitación de frecuencia de peticiones

21.8. Versionado

21.9. Gestión de errores

21.10. *Resultados de aprendizaje y criterios de evaluación asociados*

21.10.1. RA1: CE1.a, CE1.b

21.10.2. RA2

21.10.3. RA3

21.10.4. RA7: CE7.a, CE7.b, CE7.c, CE7.d, CE7.e, CE7.f, CE7.g

21.10.5. RA8: CE8.b, CE8.f, CE8.g

21.10.6. RA9: CE9.c, CE9.e, CE9.f, CE9.g

22 Contenedores *#ev2 #opcional* (est: 2020-03-09) _____

22.1. Vagrant

22.1.1. PuPHPet

22.2. Docker

22.2.1. Docker Hub

22.2.2. Dockerfiles

22.2.3. Docker Compose

- docker-compose.yml

22.3. PHPDocker

22.4. *Resultados de aprendizaje y criterios de evaluación asociados*

22.4.1. RA1: CE1.a, CE1.c, CE1.d, CE1.e

22.4.2. RA9: CE9.e, CE9.f, CE9.g

5.3. Correspondencia con resultados de aprendizaje y criterios de evaluación

El símbolo «X» representa que en esa unidad didáctica se trabaja dicho resultado de aprendizaje pero no es el elemento fundamental de evaluación.

UNIDADES DIDÁCTICAS	RA1	RA2	RA3	RA4	RA5	RA6	RA7	RA8	RA9
1. Sistemas de control de versiones I	CE1.e CE1.g			CE4.g					
2. Sistemas de control de versiones II	CE1.e CE1.g			CE4.g					
3. Introducción a la tecnología web	CE1.a CE1.b CE1.c CE1.d CE1.e CE1.g								
4. Conceptos básicos de PHP I		CE2.c CE2.d CE2.e CE2.f CE2.g CE2.h	×	CE4.g					
5. Conceptos básicos de PHP II		CE2.d CE2.e CE2.g	CE3.a CE3.b CE3.c CE3.d CE3.g	CE4.g					
6. Desarrollo de aplicaciones con PHP I	CE1.a CE1.b CE1.d CE1.f	CE2.a CE2.b CE2.c CE2.d CE2.e CE2.f CE2.g CE2.h	CE3.a CE3.b CE3.c CE3.d CE3.e CE3.f CE3.g	CE4.g	CE5.d CE5.g CE5.h				

UNIDADES DIDÁCTICAS	RA1	RA2	RA3	RA4	RA5	RA6	RA7	RA8	RA9
7. Persistencia de datos con PHP		×	×	CE4.a CE4.b CE4.c CE4.g	CE5.f CE5.g CE5.h	CE6.a CE6.b CE6.c CE6.e CE6.g			
8. Desarrollo de aplicaciones con PHP II	CE1.a			CE4.a CE4.b CE4.c CE4.d CE4.e CE4.f CE4.g	CE5.a CE5.b CE5.d CE5.f CE5.g CE5.h	CE6.a CE6.b CE6.c CE6.d CE6.e CE6.f CE6.g CE6.h			
9. Programación avanzada en PHP		×	×	CE4.g	CE5.g CE5.h				
10. Interoperabilidad	CE1.e CE1.g			CE4.g	CE5.g CE5.h				CE9.a CE9.b CE9.e
11. Introducción a Yii 2	CE1.e CE1.g			CE4.g	CE5.a CE5.b CE5.c CE5.d CE5.e CE5.g CE5.h	CE6.a CE6.b CE6.c CE6.d CE6.e CE6.f CE6.g CE6.h			CE9.b CE9.c CE9.e CE9.f CE9.g
12. Estructura de una aplicación Yii 2		×	×	CE4.g	CE5.g CE5.h				CE9.e CE9.f CE9.g
13. Gestión de peticiones en Yii 2				CE4.g	CE5.d CE5.f CE5.g CE5.h				

UNIDADES DIDÁCTICAS	RA1	RA2	RA3	RA4	RA5	RA6	RA7	RA8	RA9
14. Acceso a bases de datos en Yii2		×	×	CE4.g	CE5.f CE5.g CE5.h	CE6.a CE6.b CE6.c CE6.d CE6.e CE6.f CE6.g CE6.h			CE9.e CE9.f CE9.g
15. Creación y validación de formularios en Yii2		×	CE3.e CE3.f	CE4.g	CE5.b CE5.d CE5.h	CE6.f CE6.h		CE8.a CE8.b CE8.c CE8.d CE8.e CE8.f CE8.g	CE9.e CE9.f CE9.g
16. Visualización de datos en Yii 2		×	×	CE4.g	CE5.b CE5.c CE5.d CE5.h	CE6.f CE6.h		CE8.a CE8.b CE8.c CE8.d CE8.e CE8.f CE8.g	CE9.e CE9.f CE9.g
17. Seguridad y cacheado en Yii 2		×	×	CE4.d CE4.e CE4.g	CE5.e CE5.g CE5.h	CE6.h		×	CE9.e CE9.f CE9.g
18. Características adicionales de Yii 2		×	×	CE4.g	CE5.d CE5.g CE5.h	CE6.h		CE8.a CE8.b CE8.c CE8.d CE8.e CE8.f CE8.g	CE9.e CE9.f CE9.g
19. Calidad			CE3.g	CE4.g	CE5.g CE5.h	CE6.e CE6.h			CE9.e CE9.g

UNIDADES DIDÁCTICAS	RA1	RA2	RA3	RA4	RA5	RA6	RA7	RA8	RA9
20. Computación en la nube	CE1.a CE1.c CE1.d CE1.e				CE5.h	CE6.h			CE9.e CE9.f CE9.g
21. Servicios web con REST en Yii 2	CE1.a CE1.b	×	×				CE7.a CE7.b CE7.c CE7.d CE7.e CE7.f CE7.g	CE8.b CE8.f CE8.g	CE9.c CE9.e CE9.f CE9.g
22. Contenedores	CE1.a CE1.c CE1.d CE1.e								CE9.e CE9.f CE9.g

6. Orientaciones pedagógicas

Según queda recogido en [1], este módulo profesional contiene parte de la formación necesaria para desempeñar la función de **programación de aplicaciones de propósito general en lenguajes orientados a objetos**.

La **función** de programación de aplicaciones de propósito general en lenguajes orientados a objetos incluye aspectos como:

- El desarrollo de programas organizados en clases aplicando los principios de la programación orientada a objetos.
- La utilización de interfaces para la interacción de la aplicación con el usuario.
- La identificación, análisis e integración de librerías para incorporar funcionalidades específicas a los programas desarrollados.
- El almacenamiento y recuperación de información en sistemas gestores de bases de datos relacionales y orientados a objetos.

Las **actividades profesionales** asociadas a esta función se aplican en el desarrollo y la adaptación de programas informáticos de propósito general en lenguajes orientados a objetos.

	MODELO DE PROGRAMACIÓN ANUAL		MD7101
			VERSIÓN 0 Pág. 44 de 51

Las **líneas de actuación** en el proceso de enseñanza-aprendizaje que permiten alcanzar los objetivos del módulo profesional versarán sobre:

- La interpretación y aplicación de los principios de la programación orientada a objetos.
- La evaluación, selección y utilización de herramientas y lenguajes de programación orientados a objetos.
- La utilización de las características específicas de lenguajes y entornos de programación en el desarrollo de aplicaciones informáticas.
- La identificación de las funcionalidades aportadas por los sistemas gestores de bases de datos y su incorporación a los programas desarrollados.
- La documentación de los programas desarrollados.

7. Orientaciones metodológicas

El módulo es totalmente práctico, y el proceso de enseñanza-aprendizaje se fundamenta en la interacción continua y total de los alumnos con las herramientas software utilizadas y estudiadas a lo largo del curso.

El módulo construye el aprendizaje de forma progresiva, comenzando con el estudio de los fundamentos de la programación en un núcleo funcional sencillo para, desde ahí, ir incorporando nuevos elementos paulatinamente aumentando poco a poco la complejidad, añadiendo primero los elementos básicos de la programación imperativa, introduciendo luego las ventajas de la programación estructurada y, finalmente, incorporando los mecanismos de la programación orientada a objetos. La idea, por tanto, es hacer que cada nueva capa de complejidad se apoye en la anterior.

De la misma forma, el estudio de los datos se hará añadiendo complejidad progresiva, partiendo de los tipos de datos más básicos, introduciendo luego los tipos compuestos, estudiando las estructuras de datos clásicas y acabando con los tipos abstractos de datos. Es en ese punto donde este enfoque se encuentra con el del párrafo anterior, al hacer notar que las clases del paradigma de programación orientada a objetos son, en esencia, implementaciones de tipos abstractos de datos.

El lenguaje de programación elegido para servir de vehículo de comunicación y soporte de los conceptos aprendidos durante el curso es PHP. Los motivos que justifican tal decisión son los siguientes:

- Es un lenguaje sencillo y fácil de aprender en cuanto a sus aspectos básicos.
- Es un lenguaje interpretado y dispone de intérprete interactivo, lo que facilita el desarrollo iterativo e incremental.

- Es un lenguaje de tipado dinámico pero dispone de mecanismos opcionales de tipado estático, por lo que pasar de uno a otro resulta sencillo.
- Es un lenguaje multiparadigma, lo que permite transitar por los estilos funcional, imperativo, estructurado y orientado a objetos.
- Su sintaxis imita a la de los lenguajes *C-like*, como Java, lo que facilita la transición hacia el estudio de este último lenguaje.
- Se utiliza profusamente en 2.º curso en el módulo profesional *Desarrollo web en entorno servidor*, por lo que el tiempo invertido en su aprendizaje es tiempo bien invertido para el posterior curso.
- Es muy usado en la industria y dispone de muchas y probadas herramientas de desarrollo, así como librerías y paquetes de muy variada funcionalidad.
- Permite un acceso muy fácil a bases de datos relacionales.

No obstante lo anterior, se reconoce la importancia del lenguaje de programación Java en la industria del desarrollo de software orientado a objetos y, por consiguiente, la última parte del módulo se dedicará a su estudio, partiendo como base de todo lo aprendido en unidades anteriores apoyado en el lenguaje PHP y centrándose en las similitudes y diferencias entre ambos lenguajes. Dicho de otra forma: se hará una traducción de los conceptos codificados en PHP para estudiar cómo codificarlos en Java, y se analizarán los aspectos puramente originales de este lenguaje que no se encuentren ya en PHP y que resulten de suficiente interés como para merecer su estudio.

De manera transversal, será hincapié continuamente en el aseguramiento de la calidad del producto resultante así como del proceso de desarrollo y el código fuente desarrollado, poniendo especial interés en la metodología *TDD (Test-Driven Development)* y las pruebas automáticas.

La enseñanza se basará casi por completo en el estudio previo de los conceptos básicos y suficientes para la realización de ejercicios y supuestos de aplicación, que obliguen al alumno a enfrentarse con las herramientas software necesarias para solucionarlos.

Las actividades propuestas podrán ser individuales o grupales, aplicando donde corresponda el concepto de *programación en pareja*¹ como vehículo de colaboración y aprendizaje compartido entre varios alumnos.

Finalmente, se incentivará al alumno para que mejore su comprensión mediante el auto-aprendizaje y la elaboración propia de ejercicios y desarrollos.

¹ https://es.wikipedia.org/wiki/Programaci%C3%B3n_en_pareja

	MODELO DE PROGRAMACIÓN ANUAL	MD7101
	VERSIÓN 0	Pág. 46 de 51

8. Recursos

8.1. Hardware

- Un ordenador para cada alumno, conectado a la red local del aula y esta, a su vez, a la troncal del Centro.
- Conexión a Internet de banda ancha.
- Cañón retroproyector.

8.2. Software

- Sistema operativo GNU/Linux (Ubuntu o Debian GNU/Linux, preferentemente).
- El resto de herramientas y aplicaciones necesarias se instalarán a través de Internet a lo largo del curso.

8.3. Online

- Plataforma **Ágora**² para el seguimiento general del módulo, incluyendo distribución de material y entrega de ejercicios y exámenes.
- **GitHub** y **GitHub Classroom**³ como herramientas centralizadas para compartir código y para la gestión integral de todo elemento satélite del mismo (control de versiones, desarrollo colaborativo, incidencias, etcétera).
 - Los alumnos disponen, de forma totalmente gratuita, del **GitHub Student Developer Pack**⁴, que les ofrece servicios y herramientas con descuentos de hasta el 100 % con respecto al precio de mercado.
- **Dokuwiki**⁵ como herramienta de documentación colaborativa.

8.4. Bibliografía

8.4.1. Principal

- Apuntes y documentación *online*⁶ suministrados por el profesor.

² <http://agora.iesdonana.org>

³ <https://classroom.github.com>

⁴ <https://education.github.com/pack>

⁵ <http://wiki.iesdonana.org>

⁶ <https://pro.iesdonana.org>

	MODELO DE PROGRAMACIÓN ANUAL	MD7101	
		VERSIÓN 0	Pág. 47 de 51

- Documentación de PHP⁷.
- Documentación de Java SE⁸.

8.4.2. Complementaria

- ABELSON, H., SUSSMAN, G. J. Y SUSSMAN, J. (1996). *Structure and Interpretation of Computer Programs (2nd edition)*. Cambridge: MIT Press.
- PEÑA, R. (2003). *Diseño de programas: formalismo y abstracción (2.ª edición)*. Madrid: Prentice-Hall.
- BLANCO, J., SMITH, S., BARSOTTI, D. (2009). *Cálculo de Programas*. Córdoba (Argentina): Fa.M.A.F., Universidad Nacional de Córdoba.
- PAREJA, C., OJEDA, M., ANDEYRO, Á. L., ROSSI, C. (1997). *Desarrollo de algoritmos y técnicas de Programación en Pascal*. Madrid: Ra-Ma.

9. Atención a la diversidad

Se llevarán a cabo actividades de refuerzo o ampliación para aquellos alumnos que así lo requieran en función de las necesidades detectadas:

- Para los alumnos que muestren dificultades de aprendizaje, se propondrán **actividades de refuerzo** destinadas a afianzar los aspectos conceptuales y procedimentales en los que el alumnado presente carencias.
- Para los alumnos que muestren un mayor grado de adquisición de competencias, se propondrán **actividades de ampliación** que supongan la investigación autónoma de contenidos opcionales (aquellos marcados con la etiqueta *#opcional*).

10. Temas transversales

Con el objeto de fomentar entre los alumnos el hábito de la lectura, se plantearán actividades individuales y en grupo en las que, para su resolución, se necesite leer información de distintas fuentes escritas, como artículos, blogs, páginas web, tutoriales, etc.

La evolución experimentada por la informática en los últimos años tiene como consecuencia su influencia inevitable en todos los aspectos de las relaciones entre las personas y entre éstas

⁷ <https://php.net>

⁸ <https://docs.oracle.com/en/java/javase/>

	MODELO DE PROGRAMACIÓN ANUAL	MD7101	
		VERSIÓN 0	Pág. 48 de 51

y el entorno. Además ha demostrado ser un medio valiosísimo para la educación cualquiera que sea el ámbito en el que se use. En concreto, en cuanto a los temas transversales propuestos:

- **Educación ambiental:** La utilización de la informática, en general, y sobre todo en los negocios, hace que grandes volúmenes de información puedan ser almacenados en soportes informáticos, discos, CD, ... y enviados de unos lugares a otros a través de las redes informáticas, evitándose de esta manera el consumo de grandes cantidades de papel y, por consiguiente, la destrucción de bosques, contribuyendo de alguna manera a la preservación de los medios naturales y medioambientales.
- **Educación del consumidor:** El análisis y la utilización de diferentes herramientas informáticas favorecen la capacidad del alumnado para decidir sobre los productos informáticos que debe adquirir y utilizar de manera ventajosa.
- **Educación para la salud:** Cuando se utilizan equipos informáticos se procura que el alumnado conozcan una serie de normas de higiene y seguridad en el trabajo, así como sobre las precauciones necesarias en el empleo de los equipos. De esta manera, se intenta que el alumnado conozca los principios de la ergonomía del puesto de trabajo, para que cualquier trabajo frente al ordenador resulte lo más agradable posible y no le cause ningún problema. En este sentido, resultan de interés las instrucciones elaboradas por el Instituto Nacional de Seguridad e Higiene en el Trabajo [3].
- **Educación para la igualdad de oportunidades entre ambos sexos:** Desde este módulo contamos con elementos para concienciar al alumnado sobre la igualdad de oportunidades para alumnos y alumnas:
 - Formando grupos mixtos de trabajo.
 - Distribuyendo las tareas a realizar en la misma medida entre el alumnado de ambos sexos.
 - Haciendo que todos utilicen los mismos o equivalentes equipos.
 - Fomentando la participación de todos, sin distinciones de sexo.
- **Educación para el trabajo:** Respecto a este módulo encontramos los siguientes elementos:
 - Técnicas de trabajo en grupo: sujeción a unas reglas corporativas.
 - Colaboración de varias personas para la realización de un único trabajo.
- **Educación para la paz y la convivencia:** Se trabajan los elementos siguientes:
 - Acuerdos para la utilización de los mismos estándares en toda la comunidad internacional.
 - Respeto por las opiniones de los demás.

	MODELO DE PROGRAMACIÓN ANUAL	MD7101	
		VERSIÓN 0	Pág. 49 de 51

- Aprender a escuchar.

11. Actuaciones para desarrollar la perspectiva de género

El conocimiento de la realidad existente es el primer paso a realizar para incorporar la perspectiva de género. De esta manera, se descubrirá la existencia de situaciones de desequilibrio entre mujeres y hombres en el desempeño de la actividad docente.

La perspectiva de género es trabajada de manera transversal y permanente en todas las Unidades Didácticas que componen esta programación. El IES Doñana como organización social en aplicación de esta óptica favorece, entre otros aspectos, la detección de estereotipos y la asignación de roles y responsabilidades, la evaluación del uso y control de los recursos puestos a disposición de hombres y mujeres con la finalidad última de introducir las modificaciones y medidas correctoras necesarias para eliminar las desigualdades detectadas en cualquier ámbito de la vida del centro y particularmente dentro del aula.

En el marco de esta programación, el análisis de estas circunstancias permite identificar las diferentes necesidades, intereses y perspectivas de mujeres y hombres sobre las que diseñar estrategias que equiparen las oportunidades de ambas partes en las distintas actuaciones que lo integran. Fundamentalmente en los siguientes círculos se realizan las actuaciones:

- Profesores–profesores
- Alumnos–alumnos y
- Alumnos–profesores

Implica tener en cuenta las siguientes cuestiones:

1. Valorar la situación de partida de hombres y mujeres.
2. Analizar las necesidades y obligaciones relacionadas con la actividad cotidiana en el centro y la posición social de hombres y mujeres en el centro.
3. Velar por el cumplimiento de la condición de igualdad de género en todos los ámbitos de actuación como cuestión de justicia y responsabilidad social.

11.1. Actuaciones generales permanentes

1. Revisión del material curricular para la eliminación de la transmisión de estereotipos o modelos de conductas determinados por el género, tipo identificación cultural de funciones realizadas tradicionalmente por hombres o mujeres.
2. Detectar las desigualdades y discriminaciones de género existentes en el centro para su tratamiento/denuncia pertinente.

	MODELO DE PROGRAMACIÓN ANUAL	MD7101
	VERSIÓN 0	Pág. 50 de 51

3. Garantizar la participación equilibrada de hombres y mujeres en las distintas actividades en el aula y en el centro.
4. Velar porque el contenido gráfico y lingüístico de las acciones, materiales y dispositivos de formación y difusión carezca de cualquier carácter o pretensión discriminatoria.
5. Participación en las actividades propuestas por el Plan de Igualdad del centro articulado a través de actuaciones propias o la acción tutorial:
 - a) 25 de noviembre: Violencia de Género.
 - b) 30 de enero: Resolución de conflictos de forma pacífica. Día de la Paz.
 - c) 8 de marzo: Día de la Mujer Trabajadora.

Desde el primer momento se advertirá al alumnado que el uso del vocabulario y expresiones propias del lenguaje hablado y escrito se llevará a cabo de forma extensiva a ambos géneros, de manera que cuando hablamos del «administrador» o el «programador» lo hacemos siempre considerando que dichos roles son de aplicación a hombres y mujeres por igual. Así pues, resultará innecesario y, por tanto, se evitará el uso de fórmulas tales como «administrador o administradora», que recargan el lenguaje sin aportar información adicional. Ello además va en consonancia con lo manifestado por la Real Academia Española, al afirmar que:

*«El español dispone de un mecanismo inclusivo: el masculino gramatical, que, como término no marcado de la oposición de género, puede referirse a grupos formados de hombres y mujeres y, en contextos genéricos o inespecíficos, a personas de uno u otro sexo».*⁹

Por otra parte, en el planteamiento y realización de tareas y ejercicios, se procurará el equilibrio en cuanto a presencia de actores de ambos géneros.

Referencias

- [1] *Orden de 16 de junio de 2011*, por la que se desarrolla el currículo correspondiente al título de Técnico Superior en Desarrollo de Aplicaciones Web (págs. 130 a 133 del BOJA nº 149 del 1 de agosto)¹⁰.
- [2] *Orden de 29 de septiembre de 2010*, por la que se regula la evaluación, certificación, acreditación y titulación académica del alumnado que cursa enseñanzas de formación profesional inicial que forma parte del sistema educativo en la Comunidad Autónoma de Andalucía (BOJA nº 202 del 15 de octubre).

⁹ <https://twitter.com/RAEinforma/status/1111565711653113856>

¹⁰ <http://www.juntadeandalucia.es/boja/boletines/2011/149/d/updf/d23.pdf\T1\textbackslash#page=17>

	MODELO DE PROGRAMACIÓN ANUAL		MD7101
			VERSIÓN 0 Pág. 51 de 51

- [3] Instituto Nacional de Seguridad e Higiene en el Trabajo. *Instrucción básica para el trabajador usuario de pantallas de visualización de datos.*¹¹

¹¹ http://www.insht.es/InshtWeb/Contenidos/Documentacion/TextosOnline/Guias_Ev_Riesgos/Instruccion_Pantallas/Instruccion_basica.pdf