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

- **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.

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.

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.

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.

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 con regularidad a las clases y participar activamente en las mismas, de forma que una sistemática y frecuente falta de asistencia a clase supondrá para el alumno 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 que pudieran darse en el módulo, quedando claro que **es responsabilidad del alumno informarse activamente sobre las mismas**.

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 la parte teórica y de 4 en la parte práctica.

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 TEO ≥ 3 and PRA ≥ 4:
    EXA = TEO * 0.3 + PRA * 0.7
else:
    EXA = min(TEO, PRA)

if min(TRA, EXA) ≥ 4:
    NOTA = TRA * 0.3 + EXA * 0.7
else:
    NOTA = min(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:

$$\mathbf{NOTA} = \mathbf{EXA}$$

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:

$$\textbf{NOTA} = \textbf{TRA}$$

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 min(EV₁, EV₂, EV₃) ≥ 4:
    NOTA = EV₁ * 0.5 + EV₂ * 0.5 + EV₃ * 0.5
else:
    NOTA = min(EV₁, EV₂, EV₃)
```

Donde:

EV_1: Calificación de la primera evaluación.

EV_2: Calificación de la segunda evaluación.

EV_3: Calificación de la tercera evaluación.

El módulo se considera aprobado si **NOTA** $\geq 4{,}5$.

4.3. Medidas de recuperación

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.

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

Unidad didáctica	Inicio estimado
1. Introducción *#ev1*	2019-09-17
2. Programación funcional I *#ev1*	2019-09-24
3. Programación funcional II *#ev1*	2019-10-01
4. Programación imperativa *#ev1*	2019-10-08
5. Programación estructurada *#ev1*	2019-10-15
6. Tipos de datos *#ev1*	2019-10-22
7. Metodología de la programación *#ev1 #opcional*	2019-10-29
8. Programación modular I *#ev1*	2019-11-05
9. Entrada y salida de información *#ev1*	2019-11-12
10. Calidad I *#ev1*	2019-11-19
11. Complejidad algorítmica *#ev1 #opcional*	2019-11-26
12. Introducción al lenguaje Java *#ev2*	2020-01-15
13. Programación orientada a objetos *#ev2*	2020-01-22
14. Diseño de clases *#ev2*	2020-01-29
15. Composición, herencia y poliformismo *#ev2*	2020-02-05
16. Abstracción de datos *#ev2*	2020-02-12
17. Programación modular II *#ev2*	2020-02-19
18. Programación genérica *#ev2*	2020-02-26

Unidad didáctica	Inicio estimado
19. Estructuras de datos lineales *#ev2*	2020-03-04
20. Ordenación y búsqueda *#ev2*	2020-03-11
21. Estructuras de datos no lineales *#ev2*	2020-03-18
22. Control de excepciones *#ev2*	2020-03-25
23. Java Collections Framework *#ev3*	2020-04-13
24. Principios y patrones de diseño *#ev3*	2020-04-20
25. Calidad II *#ev3*	2020-04-27
26. Gestión de bases de datos relacionales *#ev3*	2020-05-04
27. Programación de interfaces gráficas de usuario *#ev3* *#opcional*	2020-05-11

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 **Introducción** *#ev1* *#ra1* (est: 2019–09–17)

1.1. Conceptos básicos

1.1.1. Informática

- Procesamiento automático

1.1.2. Ordenador

- Definición
- Funcionamiento básico
  - Elementos funcionales
  - Ciclo de instrucción
  - Representación de información
    - Codificación interna
      - Sistema binario
    - Codificación externa
      - ASCII
      - Unicode

1.1.3. Problema

- Generalización
- Ejemplares de un problema

- Dominio de definición
- Jerarquías de generalización

1.1.4. Algoritmo

- Definición
- Características
- Representación
  - Ordinograma
  - Pseudocódigo
- Cualidades deseables
- Computabilidad
- Corrección
- Complejidad

1.1.5. Programa

1.1.6. Lenguaje de programación

1.2. Paradigmas de programación

1.2.1. Definición

1.2.2. Imperativo

- Estructurado
- Orientado a objetos

1.2.3. Declarativo

- Funcional
- Lógico
- De bases de datos

1.3. Lenguajes de programación

1.3.1. Definición

- Sintaxis
  - Notación EBNF
- Semántica estática
- Semántica dinámica

1.3.2. Evolución histórica

1.3.3. Clasificación

- Por nivel
- Por generación
- Por propósito
- Por paradigma

1.4. Traductores

1.4.1. Definición

1.4.2. Compiladores
- Ensambladores

1.4.3. Intérpretes
- Interactivos (*REPL*)

1.5. Resolución de problemas mediante programación
1.5.1. Especificación
1.5.2. Análisis del problema
1.5.3. Diseño del algoritmo
1.5.4. Verificación
1.5.5. Estudio de la eficiencia
1.5.6. Codificación
1.5.7. Traducción y ejecución
1.5.8. Pruebas
1.5.9. Depuración
1.5.10. Documentación
1.5.11. Mantenimiento
1.5.12. Ingeniería del software
1.6. Entornos integrados de desarrollo
1.6.1. Definición
1.6.2. Editores de textos
1.6.3. Editores vs. IDE
1.6.4. Visual Studio Code

2 **Programación funcional I** *#ce1a #ce1b #ce1c #ce1e #ce1f #ce1g #ce1i #ce3f #ce3g #ev1 #ra1 #ra3 #ra6* (est: 2019–09–24)

2.1. El lenguaje de programación Python
2.1.1. Historia
2.1.2. Características principales
2.2. Modelo de ejecución
2.2.1. Modelo de sustitución
2.3. Expresiones
2.3.1. Concepto
2.3.2. Evaluación de expresiones
- Valores, expresión canónica y forma normal
- Formas normales y evaluación
- Transparencia referencial

2.3.3. Literales

2.3.4. Operaciones, operadores y operandos
- Aridad de operadores
- Paréntesis
- Prioridad de operadores
- Asociatividad de operadores

2.3.5. Funciones y métodos
- Funciones
- Igualdad de funciones
- Funciones con varios argumentos
- Composición de funciones
- Métodos

2.4. Tipos de datos

2.4.1. Concepto

2.4.2. `type`

2.4.3. Sistemas de tipos

2.4.4. Tipado fuerte vs. débil

2.4.5. Errores de tipos

2.4.6. Tipos de datos básicos
- Números
- Cadenas

2.4.7. Conversión de tipos

2.4.8. Operaciones predefinidas
- Operadores predefinidos
  - Operadores aritméticos
  - Operadores de cadenas
- Funciones predefinidas
  - Funciones matemáticas
- Métodos predefinidos

2.5. Álgebra de Boole

2.5.1. El tipo de dato *booleano*

2.5.2. Operadores relacionales

2.5.3. Operadores lógicos
- Tablas de verdad

2.5.4. Axiomas
- Traducción a Python

2.5.5. Teoremas fundamentales
- Traducción a Python

2.5.6. El operador ternario

2.6. Definiciones

2.6.1. Identificadores y ligaduras (*binding*)

- Reglas léxicas
- Constantes

2.6.2. Evaluación de expresiones con ligaduras

2.6.3. Marcos (*frames*)

2.6.4. Entorno (*environment*)

2.6.5. Tipo de un identificador

2.6.6. *Scripts*

2.6.7. Ámbito de una ligadura

2.7. Documentación interna

2.7.1. Identificadores significativos

2.7.2. Comentarios

3 **Programación funcional II** *#ce1a #ce1b #ce1c #ce3f #ce3g #ev1 #ra1 #ra3 #ra6* (est: 2019–10–01)

3.1. Abstracciones funcionales

3.1.1. Expresiones lambda

- Parámetros y argumentos
- Aplicación funcional
  - Llamadas a funciones
  - Evaluación de una aplicación funcional
- Variables ligadas y libres
- Ámbitos
  - Ámbito de una variable ligada
  - Ámbitos, marcos y entornos
  - Variables *sombreadas*
  - Renombrado de parámetros
  - Expresiones lambda y entornos
  - Evaluación de expresiones lambda con entornos
- Pureza

3.1.2. Estrategias de evaluación

- Orden de evaluación
  - Orden aplicativo
  - Orden normal

- Evaluación estricta y no estricta

3.1.3. Composición de funciones

3.2. Computabilidad

3.2.1. Funciones y procesos

3.2.2. Funciones recursivas

- Definición
- Casos base y casos recursivos
- El factorial
- Recursividad lineal
  - Procesos lineales recursivos
  - Procesos lineales iterativos
- Recursividad en árbol

3.2.3. Un lenguaje Turing-completo

3.3. Tipos de datos recursivos

3.3.1. Cadenas

3.3.2. Listas

3.3.3. Rangos

3.3.4. Conversiones entre secuencias

3.4. Funciones de orden superior

3.4.1. Concepto

3.4.2. `map`

3.4.3. `filter`

3.4.4. `reduce`

3.4.5. Listas por comprensión

4 Programación imperativa *#ce1a #ce1b #ce1c #ce3f #ce3g #ev1 #ra1 #ra3 #ra6* (est: 2019–10–08)

4.1. Modelo de ejecución

4.1.1. Máquina de estados

4.1.2. Secuencia de instrucciones

4.2. Asignación destructiva

4.2.1. Variables

4.2.2. Estado

4.2.3. Sentencia de asignación

4.2.4. Evaluación de expresiones con variables

4.2.5. Constantes

4.2.6. Tipado estático vs. dinámico

4.3. Mutabilidad

4.3.1. Tipos mutables e inmutables

- Inmutables
- Mutables

4.3.2. Alias de variables

- `id`
- `is`

4.4. Cambios de estado ocultos

4.4.1. Funciones puras

4.4.2. Funciones impuras

4.4.3. Efectos laterales

4.4.4. Transparencia referencial

4.4.5. Entrada y salida por consola

- `print`
  - El valor `None`
- `input`

4.5. Saltos

4.5.1. Incondicionales

4.5.2. Condicionales

5 **Programación estructurada** *#ce1a #ce1b #ce1c #ce3a #ce3f #ce3g #ev1 #ra1 #ra3 #ra6*
(est: 2019–10–15)

5.1. Funciones definidas por el usuario

5.1.1. Definición de funciones con nombre

5.1.2. Paso de argumentos

5.1.3. La sentencia `return`

5.1.4. Ámbito de variables

- Variables locales
- Variables globales
  - `global`
  - Efectos laterales

5.1.5. Funciones locales a funciones

- `nonlocal`

5.1.6. *Docstrings*

5.1.7. La pila de control

5.2. Aspectos teóricos de la programación estructurada

5.2.1. Programación estructurada

- Cadenas (`str`)
  - Operadores
    - Concatenación
    - Repetición
    - Indexación
    - *Slicing*
  - Funciones
  - Métodos
  - Expresiones regulares
- Listas
- Tuplas
- Rangos

6.2.2. Conjuntos (`set` y `frozenset`)

6.2.3. Diccionarios (`dict`)

6.2.4. Iterables

- Iteradores

6.3. Manipulación de tipos

6.3.1. Comprobaciones

- `str.isnumeric()`
- `str.isdigit()`

6.3.2. Conversiones

- Conversión explícita vs. implícita
- Conversión a `bool`
- Conversión a `int`
- Conversión a `float`
- Conversión a `str`

7 Metodología de la programación *#ce1a #ce1b #ce1c #ce3f #ce3g #ev1 #opcional #ra1 #ra3 #ra6* (est: 2019–10–29)

7.1. Ciclo de vida

7.2. Especificación e implementación

7.3. Verificación y validación de programas

7.3.1. Demostraciones por inducción

7.4. Programación funcional

7.4.1. Especificaciones formales

- Como cálculo

7.4.2. Derivación de programas

- Diseño recursivo
  - Recursividad final
  - Técnicas de inmersión *#opcional*

7.5. Programación imperativa

7.5.1. Especificaciones formales

- Como modificación de estados

7.5.2. Derivación de programas

- Diseño iterativo
  - Invariante de un bucle
  - Transformación de recursividad final a iterativo

7.6. El lenguaje Dafny *#opcional*

8 **Programación modular I** *#ce1a #ce1b #ce1c #ce3f #ce3g #ev1 #ra1 #ra3 #ra6* (est: 2019–11–05)

8.1. Introducción

8.1.1. Descomposición de problemas

8.2. Partes de un módulo

8.2.1. Interfaz

8.2.2. Implementación

8.2.3. Documentación interna

8.3. Importación de módulos

8.4. Paquetes

8.5. Criterios de descomposición modular

8.5.1. Abstracción

8.5.2. Ocultación de información

8.5.3. Independencia funcional

- Cohesión
- Acoplamiento

8.5.4. Reusabilidad

8.6. Diagramas de estructura

9 **Entrada y salida de información** *#ce1a #ce1b #ce1c #ce3f #ce3g #ce5a #ce5b #ce5c #ce5d #ce5e #ce6h #ce6i #ev1 #ra1 #ra3 #ra5 #ra6* (est: 2019–11–12)

9.1. Formateado de cadenas

9.2. La consola

9.2.1. Entrada desde teclado

11 **Complejidad algorítmica** *#ev1 #opcional* (est: 2019–11–26)

11.1. Introducción
11.2. Principio de invarianza
11.3. La notación asintótica *O(f(n))*
11.4. Órdenes de complejidad
11.5. Operaciones entre órdenes de complejidad
  11.5.1. Regla de la suma
  11.5.2. Regla del producto
11.6. Reglas prácticas para el cálculo de la eficiencia
11.7. Resolución de recurrencias
  11.7.1. Reducción de problemas mediante sustracción
  11.7.2. Reducción de problemas mediante división

12 **Introducción al lenguaje Java** *#ce1a #ce1b #ce1c #ce3f #ce3g #ev2 #ra1 #ra3* (est: 2020–01–15)

12.1. Introducción
12.2. Compilación vs. interpretación
  12.2.1. Máquinas reales vs. virtuales
  12.2.2. Código objeto, *bytecode* y archivos `.class`
  12.2.3. La plataforma Java
    - La máquina virtual de Java (JVM)
    - La API de Java
  12.2.4. El entorno de ejecución de Java (JRE)
    - El intérprete `java`
  12.2.5. Las herramientas de desarrollo de Java (JDK)
    - El compilador `javac`
    - El intérprete interactivo `jshell`
12.3. Características de Java
12.4. Tipado estático vs. dinámico
12.5. Tipos y valores en Java
  12.5.1. Tipos primitivos
    - Integrales
    - De coma flotante
    - Booleanos
    - Conversiones entre datos primitivos
      – *Casting*

13.3.7. Métodos
13.3.8. Constantes
13.4. Clases básicas
13.4.1. Cadenas
- Inmutables
  - `String`
- Mutables
  - `StringBuffer`
  - `StringBuilder`
  - `StringTokenizer`
- Conversión a `String`

13.4.2. *Arrays*
13.4.3. Clases *wrapper*
- Conversiones de empaquetado/desempaquetado (*boxing*/*unboxing*)

13.5. Lenguaje UML
13.5.1. Diagramas de clases
13.5.2. Diagramas de objetos
13.5.3. Diagramas de secuencia

14 **Diseño de clases** *#ce1a #ce1b #ce1c #ce2e #ce3f #ce3g #ce4a #ce4b #ce4c #ce4d #ce4e #ce4f #ce4h #ev2 #ra1 #ra2 #ra3 #ra4* (est: 2020–01–29)

14.1. Encapsulación
14.2. Propiedades
14.2.1. Visibilidad
- Pública
- Privada

14.3. Métodos
14.3.1. Visibilidad
- Pública
- Privada

14.3.2. Referencia `this`
14.3.3. Sobrecarga
14.3.4. Constructores y destructores
14.3.5. Accesores y mutadores
14.4. Constantes
14.5. Miembros estáticos
14.5.1. Constantes

- Programación a gran escala
- Programación genérica

16.2. Especificación

16.2.1. Especificaciones algebraicas

- Signatura de un TAD
  - Géneros
  - Operaciones
    - Constructoras
    - Selectoras
- Términos
- Ecuaciones

16.2.2. Construcción de especificaciones

16.2.3. Verificación con especificaciones algebraicas

16.3. Implementación

16.3.1. Pilas

16.3.2. Colas

16.3.3. Listas

17 **Programación modular II** *#ce1a #ce1b #ce1c #ce3f #ce3g #ce4i #ce4j #ev2 #ra1 #ra3 #ra4* (est: 2020–02–19)

17.1. Las clases como módulos

17.1.1. Interfaz de una clase

17.1.2. Métodos *getter* y *setter*

17.2. Interfaces

17.2.1. Definición de interfaces

17.2.2. Implementación de interfaces

17.2.3. Las interfaces como tipos

17.2.4. Conversiones entre interfaces

17.2.5. Métodos predeterminados

17.3. Paquetes y módulos

18 **Programación genérica** *#ce1a #ce1b #ce1c #ce3f #ce3g #ce6f #ev2 #ra1 #ra3 #ra6* (est: 2020–02–26)

18.1. Tipos genéricos

18.1.1. Parámetros de tipo

18.1.2. Argumentos de tipo

20.2.4. *Quicksort*

20.2.5. *Mergesort*

20.3. Tablas *Hash*

21 **Estructuras de datos no lineales** *#ce1a #ce1b #ce1c #ce3f #ce3g #ev2 #ra1 #ra3 #ra6* (est: 2020–03–18)

21.1. Árboles

21.1.1. Binarios

- Recorridos
  - Preorden
  - Inorden
  - Postorden

21.1.2. De búsqueda

21.1.3. Montículos

- Algoritmo de ordenación

21.1.4. Generales

- Recorrido en profundidad
- Recorrido en anchura

21.2. Grafos

21.2.1. Algoritmo de Dijkstra

21.2.2. Algoritmo de Floyd

22 **Control de excepciones** *#ce1a #ce1b #ce1c #ce3d #ce3f #ce3g #ev2 #ra1 #ra3* (est: 2020–03–25)

22.1. Errores y excepciones

22.2. El requisito «*captura o especifica*»

22.2.1. Tipos de excepciones

22.3. Captura y manejo de excepciones

22.3.1. Bloque `try`

22.3.2. Bloques `catch`

22.3.3. Bloque `finally`

22.4. Excepciones y signaturas

22.5. Lanzamiento de excepciones

22.5.1. Excepciones encadenadas

22.5.2. Creación de clases de excepción

22.6. Excepciones no chequeadas

22.7. Ventajas de las excepciones

23 Java Collections Framework *#ce1a #ce1b #ce1c #ce3f #ce3g #ce6a #ce6b #ce6c #ce6d #ce6e #ce6f #ev3 #ra1 #ra3 #ra6* (est: 2020–04–13)

23.1. Colecciones y *arrays*

23.2. Arquitectura

23.3. Tipos de colecciones

23.3.1. Listas ordenadas

23.3.2. Diccionarios

23.3.3. Conjuntos

23.4. Listas

23.4.1. `java.util.List`

- `java.util.ArrayList`
- `java.util.LinkedList`
- `java.util.Stack`

23.5. Colas

23.5.1. `java.util.Queue`

- `java.util.ArrayDeque`
- `java.util.PriorityQueue`
- `java.util.LinkedList`

23.5.2. `java.util.Deque`

- `java.util.ArrayDeque`
- `java.util.LinkedList`

23.6. Conjuntos

23.6.1. `java.util.Set`

- `java.util.HashSet`
- `java.util.LinkedHashSet`
- `java.util.TreeSet`

23.6.2. `java.util.SortedSet`

- `java.util.TreeSet`

23.6.3. `java.util.NavigableSet`

- `java.util.TreeSet`

23.7. Diccionarios

23.7.1. `java.util.Map`

- `java.util.HashMap`
- `java.util.LinkedHashMap`
- `java.util.TreeMap`

23.7.2. `java.util.SortedMap`

- `java.util.TreeMap`

23.7.3. `java.util.NavigableMap`

- `java.util.TreeMap`

24 **Principios y patrones de diseño** *#ce1a #ce1b #ce1c #ce3f #ce3g #ev3 #ra1 #ra3* (est: 2020–04–20)

24.1. Principios de diseño

24.1.1. Encapsulación y ocultación de información

24.1.2. Diseño orientado a interfaces

24.1.3. Principios *SOLID*

- SRP: Principio de responsabilidad única
- OCP: Principio de abierto/cerrado
- LSP: Principio de sustitución de Liskov
- ISP: Principio de segregación de la interfaz
- DIP: Principio de inversión de dependencias

24.1.4. Principio del Menor Conocimiento (o Ley de Demeter)

24.2. Patrones de diseño

24.2.1. De creación

24.2.2. Estructurales

24.2.3. De comportamiento

25 **Calidad II** *#ce1a #ce1b #ce1c #ce3f #ce3g #ev3 #ra1 #ra3* (est: 2020–04–27)

25.1. Pruebas automáticas

25.1.1. JUnit

25.2. Documentación

25.2.1. Interna

- Reglas de estilo
- Google Java Format

25.2.2. Externa

- Javadoc

26 **Gestión de bases de datos relacionales** *#ce1a #ce1b #ce1c #ce3f #ce3g #ce8a #ce8b*

#ce8c #ce8d #ce8e #ce8f #ce8g #ce8h #ce9a #ce9b #ce9c #ce9d #ce9e #ce9f #ce9g #ev3 #ra1 #ra3 #ra6 #ra8 #ra9 (est: 2020–05–04)

26.1. Controlador JDBC

26.1.1. Instalación

26.1.2. `CLASSPATH`

26.1.3. Carga

26.2. Establecimiento de conexiones

26.3. Recuperación de información

26.3.1. Ejecución de consultas

26.3.2. Selección de registros

26.3.3. Uso de parámetros

26.4. Manipulación de la información

26.4.1. Altas, bajas y modificaciones

27 **Programación de interfaces gráficas de usuario** *#ce1a #ce1b #ce1c #ce3f #ce3g #ce5f #ce5g #ce5h #ev3 #opcional #ra1 #ra3 #ra5 #ra6* (est: 2020–05–11)

27.1. JFC y Swing

27.2. Componentes de Swing

27.3. Contenedores de nivel superior

27.3.1. `JFrame`

27.3.2. `JDialog`

27.3.3. `JApplet`

27.4. `JComponent`

27.5. Componentes de texto

27.5.1. `JTextComponent`

27.6. Marcos

27.7. Etiquetas

27.8. Botones

27.9. Arquitectura de modelos de Swing

5.3. Correspondencia con resultados de aprendizaje y criterios de evaluación

El símbolo «×» 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. Introducción	×								
2. Programación funcional I	#ce1a #ce1b #ce1c #ce1e #ce1f #ce1g #ce1i		#ce3f #ce3g			×			
3. Programación funcional II	#ce1a #ce1b #ce1c		#ce3f #ce3g			×			
4. Programación imperativa	#ce1a #ce1b #ce1c		#ce3f #ce3g			×			
5. Programación estructurada	#ce1a #ce1b #ce1c		#ce3a #ce3f #ce3g			×			
6. Tipos de datos	#ce1d #ce1h		#ce3f #ce3g			#ce6g			
7. Metodología de la programación	#ce1a #ce1b #ce1c		#ce3f #ce3g			×			
8. Programación modular I	#ce1a #ce1b #ce1c		#ce3f #ce3g			×			
9. Entrada y salida de información	#ce1a #ce1b #ce1c		#ce3f #ce3g		#ce5a #ce5b #ce5c #ce5d #ce5e	#ce6h #ce6i			
10. Calidad I	#ce1a #ce1b #ce1c		#ce3f #ce3g						
11. Complejidad algorítmica									
12. Introducción al lenguaje Java	#ce1a #ce1b #ce1c		#ce3f #ce3g						
13. Programación orientada a objetos	#ce1a #ce1b #ce1c	#ce2a #ce2b #ce2c #ce2d #ce2f #ce2h #ce2i	#ce3f #ce3g			#ce6a			

Unidades didácticas	#ra1	#ra2	#ra3	#ra4	#ra5	#ra6	#ra7	#ra8	#ra9
14. Diseño de clases	#ce1a #ce1b #ce1c	#ce2e	#ce3f #ce3g	#ce4a #ce4b #ce4c #ce4d #ce4e #ce4f #ce4h					
15. Composición, herencia y poliformismo	#ce1a #ce1b #ce1c		#ce3f #ce3g	#ce4g			#ce7a #ce7b #ce7c #ce7d #ce7e #ce7f #ce7g #ce7h		
16. Abstracción de datos	#ce1a #ce1b #ce1c		#ce3f #ce3g			×			
17. Programación modular II	#ce1a #ce1b #ce1c		#ce3f #ce3g	#ce4i #ce4j					
18. Programación genérica	#ce1a #ce1b #ce1c		#ce3f #ce3g			#ce6f			
19. Estructuras de datos lineales	#ce1a #ce1b #ce1c		#ce3f #ce3g			#ce6a			
20. Ordenación y búsqueda	#ce1a #ce1b #ce1c		#ce3f #ce3g			×			
21. Estructuras de datos no lineales	#ce1a #ce1b #ce1c		#ce3f #ce3g			×			
22. Control de excepciones	#ce1a #ce1b #ce1c		#ce3d #ce3f #ce3g						
23. Java Collections Framework	#ce1a #ce1b #ce1c		#ce3f #ce3g			#ce6a #ce6b #ce6c #ce6d #ce6e #ce6f			
24. Principios y patrones de diseño	#ce1a #ce1b #ce1c		#ce3f #ce3g						

Unidades didácticas	#ra1	#ra2	#ra3	#ra4	#ra5	#ra6	#ra7	#ra8	#ra9
25. Calidad II	#ce1a #ce1b #ce1c		#ce3f #ce3g						
26. Gestión de bases de datos relacionales	#ce1a #ce1b #ce1c		#ce3f #ce3g			×		#ce8a #ce8b #ce8c #ce8d #ce8e #ce8f #ce8g #ce8h	#ce9a #ce9b #ce9c #ce9d #ce9e #ce9f #ce9g
27. Programación de interfaces gráficas de usuario	#ce1a #ce1b #ce1c		#ce3f #ce3g		#ce5f #ce5g #ce5h	×			

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.

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 basicos, introduciendo luego los tipos compuestos, estudiando los tipos abstractos de datos y acabando con la implementación de las estructuras de datos clásicas. 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.

Para servir de vehículo de comunicación y soporte de los conceptos aprendidos durante el curso se utilizarán dos lenguajes de programación: Python durante el primer trimestre y Java en los dos restantes. Los motivos que justifican el uso de python como lenguaje de introducción a la programación son los siguientes:

- Es un lenguaje sencillo y fácil de aprender en cuanto a sus aspectos básicos.
- Es el lenguaje más usado por universidades y centros de formación en cursos de introducción a la programación.
- 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.
- Es, a día de hoy, el lenguaje con mayor crecimiento en la industria del software, superando a Java y JavaScript.
- 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, por lo que resulta especialmente interesante usar dicho lenguaje en el estudio de ese paradigma de programación. Debido a ello, los trimestres segundo y tercero usarán Java, partiendo de todo lo aprendido en unidades anteriores con el lenguaje Python.

De manera transversal, se hará hincapié continuamente en el aseguramento de la calidad del producto resultante así como del proceso de desarrollo y el código fuente desarrollado, poniendo especial énfasis 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 autoaprendizaje y la elaboración propia de ejercicios y desarrollos.

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.

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

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.
- Documentación de Python[7].
- Documentación de Java SE[8].

8.4.2. Complementaria

- Abelson, H., Sussman, G. J. y Sussman, J. (1996). *Structure and Interpretation of Computer Programs (2nd edition).* Cambridge: MIT Press.

[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
[7] https://docs.python.org/3/
[8] https://docs.oracle.com/en/java/javase/

- 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.
- Joyanes Aguilar, L. (2008). *Fundamentos de programación. Algoritmos, estructuras de datos y objetos*. Aravaca: McGraw-Hill Interamericana de España.
- Van-Roy, P., Haridi, S. (2004). *Concepts, techniques, and models of computer programming*. Cambridge, Mass: MIT Press.

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 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.
  - 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.
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).

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

[9] https://twitter.com/RAEinforma/status/1111565711653113856
[10] http://www.juntadeandalucia.es/boja/boletines/2011/149/d/updf/d23.pdf\T1\textbackslash#page=17
[11] http://www.insht.es/InshtWeb/Contenidos/Documentacion/TextosOnline/Guias_Ev_Riesgos/Instruccion_Pantallas/Instruccion_basica.pdf