Programación anual

Desarrollo web en entorno servidor

Ricardo Pérez López

Curso 2019/2020

Departamento de Informática y Comunicaciones
Jefe de Departamento: Eduardo Barra Balao

Índice

1. Información general

- **Normativa de aplicación:** *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.
- **Equivalencia en créditos ECTS:** 12.
- **Código:** 0613.
- **Duración total:** 168 + 42 horas (21 semanas)
- **Carga lectiva semanal:** 8 horas + 2 horas libre configuración

2. Objetivos generales

1. La formación del módulo contribuye a alcanzar los objetivos generales de este ciclo formativo que se relacionan a continuación:

   c) Instalar módulos analizando su estructura y funcionalidad para gestionar servidores de aplicaciones.

   d) Ajustar parámetros analizando la configuración para gestionar servidores de aplicaciones.

   f) Seleccionar lenguajes, objetos y herramientas, interpretando las especificaciones para desarrollar aplicaciones web con acceso a bases de datos.

   g) Utilizar lenguajes, objetos y herramientas, interpretando las especificaciones para desarrollar aplicaciones web con acceso a bases de datos.

   h) Generar componentes de acceso a datos, cumpliendo las especificaciones, para integrar contenidos en la lógica de una aplicación Web.

   l) Utilizar herramientas y lenguajes específicos, cumpliendo las especificaciones, para desarrollar e integrar componentes software en el entorno del servidor Web.

   m) Emplear herramientas específicas, integrando la funcionalidad entre aplicaciones, para desarrollar servicios empleados en aplicaciones Web.

   n) Evaluar servicios distribuidos ya desarrollados, verificando sus prestaciones y funcionalidad, para integrar servicios distribuidos en una aplicación Web.

   ñ) Verificar los componentes de software desarrollados, analizando las especificaciones, para completar el plan de pruebas.

   q) Programar y realizar actividades para gestionar el mantenimiento de los recursos informáticos.

s) Desarrollar la creatividad y el espíritu de innovación para responder a los retos que se presentan en los procesos y organización de trabajo y de la vida personal.

t) Tomar decisiones de forma fundamentada analizando las variables implicadas, integrando saberes de distinto ámbito y aceptando los riesgos y la posibilidad de equivocación en las mismas, para afrontar y resolver distintas situaciones, problemas o contingencias.

2. La formación del módulo contribuye a alcanzar las competencias profesionales, personales y sociales de este título que se relacionan a continuación:

c) Gestionar servidores de aplicaciones adaptando su configuración en cada caso para permitir el despliegue de aplicaciones Web.

d) Gestionar bases de datos, interpretando su diseño lógico y verificando integridad, consistencia, seguridad y accesibilidad de los datos.

f) Integrar contenidos en la lógica de una aplicación web, desarrollando componentes de acceso a datos adecuados a las especificaciones.

g) Desarrollar interfaces en aplicaciones web de acuerdo con un manual de estilo, utilizando lenguajes de marcas y estándares Web.

h) Desarrollar componentes multimedia para su integración en aplicaciones web, empleando herramientas específicas y siguiendo las especificaciones establecidas.

k) Desarrollar servicios para integrar sus funciones en otras aplicaciones web, asegurando su funcionalidad.

l) Integrar servicios y contenidos distribuidos en aplicaciones web, asegurando su funcionalidad.

m) Completar planes de pruebas verificando el funcionamiento de los componentes software desarrollados, según las especificaciones.

n) Elaborar y mantener la documentación de los procesos de desarrollo, utilizando herramientas de generación de documentación y control de versiones.

ñ) Desplegar y distribuir aplicaciones web en distintos ámbitos de implantación, verificando su comportamiento y realizando modificaciones.

q) Resolver situaciones, problemas o contingencias con iniciativa y autonomía en el ámbito de su competencia, con creatividad, innovación y espíritu de mejora en el trabajo personal y en el de los miembros del equipo.

3. Resultados de aprendizaje y criterios de evaluación

1. **Selecciona las arquitecturas y tecnologías de programación Web en entorno servidor, analizando sus capacidades y características propias.**
   Criterios de evaluación:
   - *a*) Se han caracterizado y diferenciado los modelos de ejecución de código en el servidor y en el cliente Web.
   - *b*) Se han reconocido las ventajas que proporciona la generación dinámica de páginas Web y sus diferencias con la inclusión de sentencias de guiones en el interior de las páginas Web.
   - *c*) Se han identificado los mecanismos de ejecución de código en los servidores Web.
   - *d*) Se han reconocido las funcionalidades que aportan los servidores de aplicaciones y su integración con los servidores Web.
   - *e*) Se han identificado y caracterizado los principales lenguajes y tecnologías relacionados con la programación Web en entorno servidor.
   - *f*) Se han verificado los mecanismos de integración de los lenguajes de marcas con los lenguajes de programación en entorno servidor.
   - *g*) Se han reconocido y evaluado las herramientas de programación en entorno servidor.
2. **Escribe sentencias ejecutables por un servidor Web reconociendo y aplicando procedimientos de integración del código en lenguajes de marcas.**
   Criterios de evaluación:
   - *a*) Se han reconocido los mecanismos de generación de páginas Web a partir de lenguajes de marcas con código embebido.
   - *b*) Se han identificado las principales tecnologías asociadas.
   - *c*) Se han utilizado etiquetas para la inclusión de código en el lenguaje de marcas.
   - *d*) Se ha reconocido la sintaxis del lenguaje de programación que se ha de utilizar.
   - *e*) Se han escrito sentencias simples y se han comprobado sus efectos en el documento resultante.
   - *f*) Se han utilizado directivas para modificar el comportamiento predeterminado.
   - *g*) Se han utilizado los distintos tipos de variables y operadores disponibles en el lenguaje.
   - *h*) Se han identificado los ámbitos de utilización de las variables.

3. **Escribe bloques de sentencias embebidos en lenguajes de marcas, seleccionando y utilizando las estructuras de programación.**
   Criterios de evaluación:
   a) Se han utilizado mecanismos de decisión en la creación de bloques de sentencias.
   b) Se han utilizado bucles y se ha verificado su funcionamiento.
   c) Se han utilizado «arrays» para almacenar y recuperar conjuntos de datos.
   d) Se han creado y utilizado funciones.
   e) Se han utilizado formularios Web para interactuar con el usuario del navegador Web.
   f) Se han empleado métodos para recuperar la información introducida en el formulario.
   g) Se han añadido comentarios al código.
4. **Desarrolla aplicaciones Web embebidas en lenguajes de marcas analizando e incorporando funcionalidades según especificaciones.**
   Criterios de evaluación:
   a) Se han identificado los mecanismos disponibles para el mantenimiento de la información que concierne a un cliente Web concreto y se han señalado sus ventajas.
   b) Se han utilizado sesiones para mantener el estado de las aplicaciones Web.
   c) Se han utilizado «cookies» para almacenar información en el cliente Web y para recuperar su contenido.
   d) Se han identificado y caracterizado los mecanismos disponibles para la autentificación de usuarios.
   e) Se han escrito aplicaciones que integren mecanismos de autentificación de usuarios.
   f) Se han realizado adaptaciones a aplicaciones Web existentes como gestores de contenidos u otras.
   g) Se han utilizado herramientas y entornos para facilitar la programación, prueba y depuración del código.
5. **Desarrolla aplicaciones Web identificando y aplicando mecanismos para separar el código de presentación de la lógica de negocio.**
   Criterios de evaluación:
   a) Se han identificado las ventajas de separar la lógica de negocio de los aspectos de presentación de la aplicación.
   b) Se han analizado tecnologías y mecanismos que permiten realizar esta separación y sus características principales.

c) Se han utilizado objetos y controles en el servidor para generar el aspecto visual de la aplicación Web en el cliente.

d) Se han utilizado formularios generados de forma dinámica para responder a los eventos de la aplicación Web.

e) Se han identificado y aplicado los parámetros relativos a la configuración de la aplicación Web.

f) Se han escrito aplicaciones Web con mantenimiento de estado y separación de la lógica de negocio.

g) Se han aplicado los principios de la programación orientada a objetos.

h) Se ha probado y documentado el código.

6. **Desarrolla aplicaciones de acceso a almacenes de datos, aplicando medidas para mantener la seguridad y la integridad de la información.**
   Criterios de evaluación:

   a) Se han analizado las tecnologías que permiten el acceso mediante programación a la información disponible en almacenes de datos.

   b) Se han creado aplicaciones que establezcan conexiones con bases de datos.

   c) Se ha recuperado información almacenada en bases de datos.

   d) Se ha publicado en aplicaciones Web la información recuperada.

   e) Se han utilizado conjuntos de datos para almacenar la información.

   f) Se han creado aplicaciones Web que permitan la actualización y la eliminación de información disponible en una base de datos.

   g) Se han utilizado transacciones para mantener la consistencia de la información.

   h) Se han probado y documentado las aplicaciones.

7. **Desarrolla servicios Web analizando su funcionamiento e implantando la estructura de sus componentes.**
   Criterios de evaluación:

   a) Se han reconocido las características propias y el ámbito de aplicación de los servicios Web.

   b) Se han reconocido las ventajas de utilizar servicios Web para proporcionar acceso a funcionalidades incorporadas a la lógica de negocio de una aplicación.

   c) Se han identificado las tecnologías y los protocolos implicados en la publicación y utilización de servicios Web.

d) Se ha programado un servicio Web.

e) Se ha creado el documento de descripción del servicio Web.

f) Se ha verificado el funcionamiento del servicio Web.

g) Se ha consumido el servicio Web.

8. **Genera páginas Web dinámicas analizando y utilizando tecnologías del servidor Web que añadan código al lenguaje de marcas.**
Criterios de evaluación:

a) Se han identificado las diferencias entre la ejecución de código en el servidor y en el cliente Web.

b) Se han reconocido las ventajas de unir ambas tecnologías en el proceso de desarrollo de programas.

c) Se han identificado las librerías y las tecnologías relacionadas con la generación por parte del servidor de páginas Web con guiones embebidos.

d) Se han utilizado estas tecnologías para generar páginas Web que incluyan interacción con el usuario en forma de advertencias y peticiones de confirmación.

e) Se han utilizado estas tecnologías, para generar páginas Web que incluyan verificación de formularios.

f) Se han utilizado estas tecnologías para generar páginas Web que incluyan modificación dinámica de su contenido y su estructura.

g) Se han aplicado estas tecnologías en la programación de aplicaciones Web.

9. **Desarrolla aplicaciones Web híbridas seleccionando y utilizando librerías de código y repositorios heterogéneos de información.**
Criterios de evaluación:

a) Se han reconocido las ventajas que proporciona la reutilización de código y el aprovechamiento de información ya existente.

b) Se han identificado librerías de código y tecnologías aplicables en la creación de aplicaciones Web híbridas.

c) Se ha creado una aplicación Web que recupere y procese repositorios de información ya existentes.

d) Se han creado repositorios específicos a partir de información existente en Internet y en almacenes de información.

e) Se han utilizado librerías de código para incorporar funcionalidades específicas a una aplicación Web.

f) Se han programado servicios y aplicaciones Web utilizando como base información y código generados por terceros.

g) Se han probado, depurado y documentado las aplicaciones generadas.

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.

4.1. Valoración general de los contenidos

Los contenidos se ponderarán en base a los siguientes porcentajes:

Trabajos, actividades y ejercicios (casa, clase, grupo) (**TR**)	30 %
Pruebas evaluativas (**EX**)	70 %

Si, por algún motivo, no se pudiera evaluar uno de los dos apartados anteriores (**TR**o **EX**), 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 **TR**) 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 **EX**), 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, no es posible evaluar el segundo trimestre por separado del primero, por lo que los exámenes evaluarán todos los contenidos trabajados hasta el momento desde el comienzo del curso, de forma que:

- **En la primera evaluación**, el examen evaluará los contenidos trabajados en el primer trimestre.
- **En la segunda evaluación**, el examen evaluará los contenidos trabajados en el segundo trimestre pero pudiendo incluir aspectos del primer trimestre.

4.2.1. Calificaciones parciales

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

Algoritmo 1 (Cálculo de la calificación parcial)

```
if (mín(TR, EX) ≥ 4) {
    NOTA = TR * 0.3 + EX * 0.7;
} else {
    NOTA = mín(TR, EX);
}
```

Donde:

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

EX: Calificación del examen correspondiente a ese trimestre, valorada del 0 al 10.

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 TR) dignos de calificación, entonces los porcentajes se redistribuirán de forma que la calificación de la evaluación parcial correspondiente resultará ser la nota del examen (apartado EX), por lo que quedará simplemente de la siguiente forma:

$$\textbf{NOTA} = \textbf{EX}$$

Asimismo, si durante el trimestre en cuestión no se realizaran exámenes (apartado EX), entonces los porcentajes se redistribuirán de forma que la calificación de la evaluación parcial correspondiente resultará ser la nota de los trabajos, actividades y/o ejercicios (apartado TR), por lo que quedará simplemente de la siguiente forma:

$$\textbf{NOTA} = \textbf{TR}$$

4.2.2. Calificación final

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

Algoritmo 2 (Cálculo de la calificación final)

```
if (mín(EV₁, EV₂) ≥ 4) {
    NOTA = TR * 0.3 + EX * 0.7;
} else {
    NOTA = mín(TR, EX);
}
```

Donde:

EV_1: Calificación de la primera evaluación.

EV_2: Calificación de la segunda evaluación.

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

Además de lo anterior, 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:

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

2.d) Se ha reconocido la sintaxis del lenguaje de programación que se ha de utilizar.

2.g) Se han utilizado los distintos tipos de variables y operadores disponibles en el lenguaje.

2.h) Se han identificado los ámbitos de utilización de las variables.

3.a) Se han utilizado mecanismos de decisión en la creación de bloques de sentencias.

3.b) Se han utilizado bucles y se ha verificado su funcionamiento.

3.c) Se han utilizado «arrays» para almacenar y recuperar conjuntos de datos.

3.d) Se han creado y utilizado funciones.

3.e) Se han utilizado formularios Web para interactuar con el usuario del navegador Web.

3.f) Se han empleado métodos para recuperar la información introducida en el formulario.

3.g) Se han añadido comentarios al código.

4.b) Se han utilizado sesiones para mantener el estado de las aplicaciones Web.

4.e) Se han escrito aplicaciones que integren mecanismos de autentificación de usuarios.

5.f) Se han escrito aplicaciones Web con mantenimiento de estado y separación de la lógica de negocio.

5.g) Se han aplicado los principios de la programación orientada a objetos.

6.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.

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.

4.2.3. Medidas de recuperación

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 pendiente algún contenido 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).

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.

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.

1. **Sistemas de control de versiones I** #ev1 (due: 2019-09-16)
    1.1. Preparación del entorno de desarrollo
        1.1.1. Instalación automatizada
            1.1.1.1. Acciones previas
                1.1.1.1.1. Instalar `git`
                    1.1.1.1.1.1. Crear cuenta en GitHub
                    1.1.1.1.1.2. Solicitar el Student Developer Pack
                1.1.1.1.2. Usar https://github.com/ricpelo/conf y seguir las instrucciones del `README.md`
            1.1.1.2. Terminal
                1.1.1.2.1. `Zsh`
                1.1.1.2.2. Oh My Zsh
                1.1.1.2.3. `less`
            1.1.1.3. Navegador

1.1.1.4. Editores de texto

1.1.1.4.1. Vim y less

1.1.1.4.2. Atom

1.1.1.4.2.1. Instalación

1.1.1.4.2.2. Configuración

1.1.1.4.2.3. Paquetes

1.1.1.4.3. Alternativa: PhpStorm

1.1.1.5. DokuWiki #opcional

1.1.1.5.1. Elaboración de documentación

1.1.1.5.2. La wiki como sistema de control de versiones

1.1.1.5.3. La wiki como herramienta colaborativa

1.1.1.6. Ejercicios

1.1.1.6.1. Indica en la wiki tu nombre completo junto a tu nombre de usuario en GitHub.

1.1.1.6.2. Sigue el tutorial de `vimtutor` y envía el archivo resultante.

1.1.1.6.3. Escoge un paquete del repositorio de paquetes de Atom e indica en la wiki su nombre, su funcionamiento básico, un enlace al paquete dentro del repositorio y por qué te ha resultado interesante.

1.1.1.6.4. Escoge una extensión de Google Chrome e indica en la wiki su nombre, su funcionamiento básico, un enlace a la extensión y por qué te ha resultado interesante.

1.1.1.6.5. Indica qué editor de textos sería más apropiado usar en cada una de las situaciones siguientes:

1.1.1.6.5.1. Programar en PHP.

1.1.1.6.5.2. Escribir un pequeño script.

1.1.1.6.5.3. Cambiar un archivo de configuración del sistema.

1.1.1.6.5.4. Editar un archivo situado en otro equipo a través de la red. <

1.1.1.7. Primeros pasos

1.1.1.7.1. `config`

1.1.1.7.2. `git-config.sh`

1.1.1.7.3. `init`

1.1.1.7.4. `add`

1.1.1.7.5. `commit`

1.1.1.7.5.1. Con la opción `-m`

1.1.1.7.5.2. Sin la opción `-m`

1.1.1.7.6. `checkout` (descartar cambios)

1.1.1.7.7. `reset`

1.1.1.7.8. `.gitignore`

1.1.1.8. Estado

1.1.1.8.1. `status`

1.1.1.8.2. `log`

1.1.1.8.3. Alias `lg`

1.1.1.8.4. `show`

1.1.1.8.5. `diff`

1.1.1.8.5.1. `git diff`

1.1.1.8.5.2. `git diff --staged`

1.1.1.8.5.3. `git diff <commit>`

1.1.1.8.5.4. `git diff inicial..final`

1.1.1.8.6. Referencias

1.1.1.8.6.1. `HEAD` y `master`

1.1.1.8.6.2. `237ab45^+`

1.1.1.8.6.3. `237ab45 1`

1.1.1.9. La máquina del tiempo

1.1.1.9.1. `checkout` (mover el `HEAD`)

1.1.1.9.2. `revert`

1.1.1.9.3. `reset`

1.1.1.9.4. `tag`

1.1.1.9.5. `--amend`

1.1.1.10. Borrar y mover

1.1.1.10.1. `rm`

1.1.1.10.2. `mv`

1.1.1.11. Git y los directorios

1.1.1.12. Ejercicios

1.1.1.12.1. Sigue el tutorial de Git Immersion desde el LAB3 hasta el LAB20, comprime y envía el directorio resultante.

1.1.1.13. Meta

1.1.1.13.1. Objetivos de la unidad

1.1.1.13.1.1. Reconocer la importancia y la necesidad de usar un sistema de control de versiones durante el desarrollo de software.

1.1.1.13.1.2. Reconocer la utilidad de un sistema de control de versiones en tareas tan diversas como documentación, copias de seguridad, colaboración, despliegue de aplicaciones, etc.

1.1.1.13.1.3. Entender la diferencia entre sistemas de control de versiones centralizados y distribuidos, y cómo estos últimos superan abiertamente a los primeros.

1.1.1.13.1.4. Reconocer a Git como un sistema de control de versiones distribuido.

1.1.1.13.1.5. Reconocer la importancia que tiene Git en el panorama actual de desarrollo de software.

1.1.1.13.1.6. Entender los conceptos de repositorio, directorio de trabajo, stage, commit, log.

1.1.1.13.1.7. Aprender el funcionamiento básico de Git en un repositorio local.

1.1.1.13.1.8. Aprender a moverse a través del tiempo por los commits de un repositorio Git.

1.1.1.13.1.9. Aprender a corregir commits creando nuevos commits.

1.1.1.13.2. Resultados de aprendizaje y criterios de evaluación asociados

1.1.1.13.2.1. RA1

1.1.1.13.2.1.1. CE1.e

1.1.1.13.2.1.2. CE1.g

1.1.1.13.2.2. RA4

1.1.1.13.2.2.1. CE4.g

2. **Sistemas de control de versiones II** #ev1 (due: 2019-09-23)
    2.1. Ramas locales
        2.1.1. `branch`
        2.1.2. `merge`
            2.1.2.1. Estrategia *fast-forward*
            2.1.2.2. Estrategia *recursive* o del padre múltiple
            2.1.2.3. `--no-ff`
        2.1.3. Resolución de conflictos
        2.1.4. `rebase`
        2.1.5. Ejercicios
            2.1.5.1. Sigue el tutorial de Git Immersion desde el LAB24 hasta el LAB35, comprime y envía el directorio resultante.
    2.2. Ramas remotas
        2.2.1. Orígenes remotos
            2.2.1.1. Directorios locales
            2.2.1.2. Servidores remotos con repositorios compartidos
            2.2.1.3. `remote [add|show] origin`
        2.2.2. Flujo de trabajo básico
            2.2.2.1. `push`
                2.2.2.1.1. Ramas de seguimiento (*tracking branch*)
            2.2.2.2. `clone`
                2.2.2.2.1. ¿Qué significa `origin/HEAD`?
            2.2.2.3. `fetch`
            2.2.2.4. `pull`
        2.2.3. Eliminar ramas remotas #no_impartido
        2.2.4. Etiquetas remotas #no_impartido
            2.2.4.1. `git push origin mi_etiqueta`
            2.2.4.2. `git push --tags`
            2.2.4.3. `git push --delete origin mi_etiqueta`

2.2.5. Ejercicios

2.2.5.1. Sigue el tutorial de Git Immersion desde el LAB36 hasta el LAB49, comprime y envía el directorio resultante.

2.3. GitHub

2.3.1. El flujo de trabajo de GitHub

2.3.2. Pull requests

2.3.2.1. Comentarios generales y comentarios en línea

2.3.2.2. Revisiones de cambios

2.3.2.2.1. Crear y solicitar revisiones

2.3.2.3. Arreglar una PR

2.3.2.4. Cerrar una PR

2.3.2.4.1. `git remote prune origin`

2.3.3. Issues

2.3.4. Releases #no_impartido

2.3.5. Forks

2.3.6. GitHub Education

2.3.6.1. GitHub Classroom

2.4. Ejercicios

2.4.1. Sigue el tutorial de Hello World e indica el repositorio resultante.

2.4.2. Sigue el tutorial de Meghan Nelson e indica el repositorio resultante.

2.5. Meta

2.5.1. Objetivos de la unidad

2.5.2. Resultados de aprendizaje y criterios de evaluación asociados

2.5.2.1. RA1

2.5.2.1.1. CE1.e

2.5.2.1.2. CE1.g

2.5.2.2. RA4

2.5.2.2.1. CE4.g

3. **Introducción a la tecnología web** #ev1 (due: 2019-09-30)

3.1. Introducción al desarrollo web

3.1.1. Conceptos básicos

3.1.1.1. Navegadores y servidores web

3.1.1.2. Agentes de usuario

3.1.1.3. Web estática vs. dinámica

3.1.1.4. Estructura vs. contenido

3.1.1.5. Arquitectura multinivel

3.1.2. Ejemplos de aplicaciones web

3.1.2.1. Redes sociales: Facebook, Twitter…

3.1.2.2. Comercio electrónico: Amazon, eBay…

3.1.2.3. Administración electrónica…

3.1.2.4. Portales

3.1.2.5. ERP, CRM

3.1.3. Tecnologías de desarrollo de aplicaciones web

3.1.3.1. .NET

3.1.3.2. Java

3.1.3.3. Ruby/Rails

3.1.3.4. Python/Django

3.1.3.5. PHP

3.1.3.6. El Kung-Fu de la Programación

3.1.3.6.1. Odoo

3.1.3.6.2. PrestaShop

3.1.3.6.3. Drupal

3.1.3.6.4. WordPress

3.2. Arquitectura cliente/servidor

3.3. HTML 5 básico (recordatorio de primer curso)

3.4. Protocolo HTTP

3.4.1. URIs

3.4.1.1. URL encoding

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

3.4.3. Métodos: GET, POST

3.4.4. Versiones

3.4.4.1. HTTP/1.0

3.4.4.2. HTTP/1.1

3.4.5. Cabeceras HTTP

3.4.6. Códigos de estado

3.4.7. Experimentos

3.4.7.1. `telnet` (a un servidor)

3.4.7.2. `netcat` (desde un navegador)

3.4.7.3. `curl -i -XPOST "http://..." | pygmentize -l http` #no_impartido

3.4.7.4. `http` #no_impartido

3.4.7.5. Google Chrome Developer Tools #no_impartido

3.4.8. Envío de datos al servidor

3.4.8.1. Mediante GET

3.4.8.2. Mediante POST

3.4.8.3. 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. Meta

3.6.1. Objetivos de la unidad

3.6.2. Resultados de aprendizaje y criterios de evaluación asociados

3.6.2.1. RA1

3.6.2.1.1. CE1.a

3.6.2.1.2. CE1.d

3.6.2.1.3. CE1.e

3.6.2.1.4. CE1.g

4. **Conceptos básicos de PHP I** #ev1 (due: 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
            4.2.2.1. Comando `echo`
        4.2.3. Expresiones, operadores y funciones
    4.3. Funcionamiento del intérprete
        4.3.1. Ejecución
            4.3.1.1. Por lotes
            4.3.1.2. Interactiva
                4.3.1.2.1. `php -a`
                4.3.1.2.2. 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`)
            4.5.1.1. Operadores lógicos
        4.5.2. Numéricos
            4.5.2.1. Enteros (`int`)

5. **Conceptos básicos de PHP II** #ev1 (due: 2019-10-14)
    5.1. Flujo de control
        5.1.1. Estructuras de control
            5.1.1.1. Sintaxis alternativa
        5.1.2. Inclusión de scripts
            5.1.2.1. `include`, `require`
            5.1.2.2. `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
            5.3.1.1. Acceso, modificación y agregación
        5.3.2. Funciones de manejo de arrays
            5.3.2.1. Ordenación de arrays
            5.3.2.2. `print_r()`
            5.3.2.3. `'+'` vs. `array_merge()`
            5.3.2.4. `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
            5.4.1.1. Paso de argumentos por valor y por referencia
            5.4.1.2. Argumentos por defecto
        5.4.2. Ámbito de variables
            5.4.2.1. Ámbito simple al archivo
            5.4.2.2. Variables locales

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. Programación orientada a objetos básica

6.4.1. Objetos

6.4.1.1. `new`

6.4.1.2. `instanceof`

6.4.2. Referencias

6.4.2.1. Asignación por referencia (`=&`)

6.4.3. Clonación de objetos

6.4.4. Comparación de objetos

6.4.5. Propiedades

6.4.5.1. Predeterminadas

6.4.5.2. Dinámicas

6.4.6. Métodos

6.4.7. Constantes

6.4.7.1. 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`

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

7.1.1. Clase `PDO`

7.1.1.1. `__construct(string $dsn [, string $username [, string $password [, array $options ]]])`

7.1.1.2. `PDOStatement query(string $statement)`

7.1.1.3. `int exec(string $statement)`

7.1.1.4. `PDOStatement prepare(string $statement [, array $driver_options = array() ])`

7.1.2. Clase `PDOStatement`

7.1.2.1. `mixed fetch([ int $fetch_style ])`

7.1.2.2. `mixed fetchAll([ int $fetch_style ])`

7.1.2.3. `mixed fetchColumn([ int $column_number = 0 ])`

7.1.2.4. `bool execute ([ array $input_parameters ])`

7.1.2.5. `int rowCount(void)`

7.1.3. Correspondencias de tipos entre SQL y PHP

7.1.4. Transacciones

7.1.4.1. `$pdo->beginTransaction();`

7.1.4.2. `$pdo->commit();`

7.1.4.3. `$pdo->rollBack();`

7.2. Cookies

7.2.1. `setcookie()`

7.2.2. Ejemplos de uso

7.3. Sesiones

7.3.1. Iniciar una sesión

7.3.1.1. `session_start()`

7.3.2. Usar una sesión

7.3.2.1. `$_SESSION`

7.3.2.2. Ejemplos de uso

7.3.3. Terminar una sesión

7.3.3.1. `session_destroy()`

7.5.2.5.4. CE6.d

7.5.2.5.5. CE6.e

7.5.2.5.6. CE6.f

7.5.2.5.7. CE6.g

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

8.1. CRUD

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`

8.4. Seguridad básica

8.4.1. Filtrar la entrada, escapar la salida

8.4.2. Cross-Site Scripting (XSS)

8.4.2.1. No persistente

8.4.2.2. Persistente

8.4.2.3. Escapado de la salida

8.4.2.3.1. `htmlspecialchars()`

8.4.2.3.2. HTML Purifier #no_impartido

8.4.3. Filtrado de la entrada

8.4.3.1. Cómo *NO* se debe hacer

8.4.3.2. Extensión Filter

8.4.3.2.1. `filter_input()`, `filter_has_var()`, `filter_var()`

8.4.3.2.2. Filtros de validación y saneado

8.4.3.3. Expresiones regulares (PCRE)

9. **Programación avanzada en PHP** #ev1 (due: 2019-11-11)

9.1. Diseño de aplicaciones orientadas a objetos

9.1.1. Clases

9.1.2. Propiedades

9.5.2.2. RA3

9.5.2.3. RA4

9.5.2.4. RA5

9.5.2.4.1. CE5.g

10. **Interoperabilidad** #ev1 (due: 2019-11-18)

10.1. Versionado semántico

10.2. Composer

10.2.1. Paquetes

10.2.2. Packagist

10.2.3. Dependencias

10.2.3.1. `composer.json` y `composer.lock`

10.2.4. Versiones y restricciones

10.2.4.1. Versión exacta

10.2.4.2. Rango (>, >=, <, <=, `!=`, , , , `||`)

10.2.4.3. Guión (-)

10.2.4.4. Asterisco (*)

10.2.4.5. Tilde ( )

10.2.4.6. Gorrito (^+)

10.2.4.7. Nombres de rama

10.2.4.7.1. `dev-master`

10.2.4.7.2. `5.1.x-dev`

10.2.4.8. Estabilidad mínima

10.2.4.9. Comprobador online de restricciones

10.2.5. Comandos básicos

10.2.5.1. `require`

10.2.5.2. `install`

10.2.5.3. `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. Ejercicios

10.7.1. De versionado semántico

10.7.2. De versiones y restricciones

10.7.3. De uso básico de Composer

10.7.4. De buscar paquetes en Packagist que tengan una funcionalidad concreta y usarlos en un ejemplo

10.8. Meta

10.8.1. Objetivos de la unidad

10.8.2. Resultados de aprendizaje y criterios de evaluación asociados

10.8.2.1. RA1

10.8.2.1.1. CE1.e

10.8.2.1.2. CE1.g

10.8.2.2. RA9

10.8.2.2.1. CE9.a

10.8.2.2.2. CE9.b

10.8.2.2.3. CE9.e

11. **Introducción a Yii 2** #ev1 (due: 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

11.4.2.1. Instalación mediante Composer

11.4.3. Plantillas de proyecto

11.4.3.1. Plantilla básica vs. avanzada

11.4.3.2. 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. Ejercicios

11.11.1. Definiciones: framework, microframework, librería (semejanzas y diferencias). Modelo, vista y controlador. Rutas y ejemplos de rutas.

11.11.2. ¿Qué es una plantilla de proyecto? Diferencias entre la plantilla básica y la avanzada.

11.11.3. ¿Qué herramientas de desarrollo tenemos para desarrollar una aplicación en Yii 2 y para qué sirven?

11.11.4. Crear un proyecto desde cero, paso a paso.

11.11.5. Escribir programas sencillos, con formularios, validaciones sencillas y acceso a bases de datos.

11.12. Meta

11.12.1. Objetivos de la unidad

11.12.2. Resultados de aprendizaje y criterios de evaluación asociados

11.12.2.1. RA1

11.12.2.1.1. CE1.e

11.12.2.1.2. CE1.g

11.12.2.2. RA4

11.12.2.3. RA5

11.12.2.3.1. CE5.a

11.12.2.3.2. CE5.b

11.12.2.3.3. CE5.e

11.12.2.3.4. CE5.g

11.12.2.4. RA9

11.12.2.4.1. CE9.b

11.12.2.4.2. CE9.c

11.12.2.4.3. CE9.e

11.12.2.4.4. CE9.f

11.12.2.4.5. CE9.g

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

12.1. A pequeña escala

12.1.1. Componentes

12.1.1.1. La clase `yii\base\BaseObject`

12.1.1.1.1. Propiedades

12.1.1.1.2. Configuraciones

12.1.1.1.2.1. Asignación masiva

12.1.1.1.2.2. Creación de nuevas instancias

12.1.1.1.2.3. Normas de creación de componentes

12.1.1.1.2.4. Diferencias entre `new` y `Yii::createObject()`

12.1.1.2. La clase `yii\base\Component`

12.1.1.2.1. Eventos

12.1.1.2.1.1. De instancia

12.1.1.2.1.1.1. Eventos de instancia

12.1.1.2.1.1.2. Manejadores de eventos de instancia

12.1.1.2.1.2. De clase

12.1.1.2.1.2.1. Eventos de clase

12.1.1.2.1.2.2. Manejadores de eventos de clase

12.1.1.2.2. 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

12.3.8. ¿Qué es un script de entrada? ¿Cuántos scripts de entrada hay en una aplicación Yii 2? ¿Dónde se encuentran?

12.3.9. ¿Qué es el objeto Aplicación? ¿Qué es un componente de aplicación?

12.3.10. ¿Qué es un controlador? Diferencia entre controlador web y controlador de consola. ¿Dónde se almacenan? ¿Qué es una acción? Ejemplo de ambos tipos.

12.3.11. ¿Qué es un modelo? ¿Dónde se almacenan?

12.3.12. ¿Qué es una vista? ¿Dónde se almacenan? Ejemplo.

12.3.13. ¿Qué es un filtro?

12.4. Meta

12.4.1. Objetivos de la unidad

12.4.2. Resultados de aprendizaje y criterios de evaluación asociados

12.4.2.1. RA2

12.4.2.2. RA3

12.4.2.3. RA4

12.4.2.4. RA5

12.4.2.4.1. CE5.g

12.4.2.5. RA9

12.4.2.5.1. CE9.e

12.4.2.5.2. CE9.f

12.4.2.5.3. CE9.g

13. **Gestión de peticiones en Yii 2** #ev2 (due: 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. Ejercicios

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

14.4.2.1. RA2

14.4.2.2. RA3

14.4.2.3. RA4

14.4.2.4. RA5

14.4.2.5. RA6

14.4.2.5.1. CE6.a

14.4.2.5.2. CE6.b

14.4.2.5.3. CE6.c

14.4.2.5.4. CE6.d

14.4.2.5.5. CE6.e

14.4.2.5.6. CE6.f

14.4.2.5.7. CE6.g

14.4.2.6. RA9

14.4.2.6.1. CE9.e

14.4.2.6.2. CE9.f

14.4.2.6.3. CE9.g

15. **Creación y validación de formularios en Yii 2** #ev2 (due: 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

15.2.1.1. Validadores principales

15.2.1.2. `skipOnEmpty`

15.2.1.3. `skipOnError`

15.2.1.4. Personalizar mensajes de error

15.2.1.5. Validación condicional

15.2.1.6. Filtrado (saneado) de datos

15.2.1.7. Manejo de entradas vacías

16.2. Paginación

16.2.1. `yii\data\Pagination`

16.2.2. Entrada

16.2.2.1. `totalCount`

16.2.2.2. `pageSize`

16.2.2.3. `page`

16.2.3. Salida

16.2.3.1. `limit`

16.2.3.2. `offset`

16.2.4. `yii\widgets\LinkPager`

16.3. Ordenación

16.3.1. `yii\data\Sort`

16.3.2. Entrada

16.3.2.1. `attributes`

16.3.2.2. `sort`

16.3.3. Salida

16.3.3.1. `orders`

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

16.4. Proveedores de datos

16.4.1. Entrada

16.4.1.1. `pagination`

16.4.1.2. `sort`

16.4.2. Salida

16.4.2.1. `models`

16.4.2.2. `count`

16.4.2.3. `totalCount`

16.4.3. `ActiveDataProvider`

16.4.3.1. `query`

16.4.4. `SqlDataProvider`

16.4.4.1. `sql`

16.4.4.2. `params`

16.4.4.3. `totalCount`

16.4.5. `ArrayDataProvider`

16.4.5.1. `allModels`

16.5. Widgets de datos

16.5.1. `DetailView`

16.5.2. `ListView`

16.5.3. `GridView`

16.5.3.1. `dataProvider`

16.5.3.2. `columns`

16.5.3.2.1. `DataColumn`

16.5.3.2.2. `SerialColumn`

16.5.3.2.3. `ActionColumn`

16.5.3.3. Ordenación de columnas

16.5.3.4. Filtrado de datos

16.5.3.4.1. `yii\grid\GridView::$filterModel`

16.5.3.5. Relaciones

16.5.3.5.1. Ordenación con relaciones

16.5.3.5.2. Filtrado con relaciones

16.5.3.5.2.1. `yii\db\ActiveRecord::getAttribute($name)`

16.5.4. El problema de las fechas/horas/instantes

16.5.5. Otros widgets no oficiales

16.5.5.1. Krajee Yii Extensions

16.5.5.1.1. `yii2-datecontrol`

16.5.5.1.2. `yii2-number`

16.6. Scripts de cliente

16.7. Temas #opcional

16.8. Meta

16.8.1. Objetivos de la unidad

16.8.2. Resultados de aprendizaje y criterios de evaluación asociados

16.8.2.1. RA2

16.8.2.2. RA3

16.8.2.3. RA4

16.8.2.4. RA5

16.8.2.4.1. CE5.b

16.8.2.4.2. CE5.c

16.8.2.4.3. CE5.d

16.8.2.5. RA8

16.8.2.5.1. CE8.a

16.8.2.5.2. CE8.b

16.8.2.5.3. CE8.c

16.8.2.5.4. CE8.d

16.8.2.5.5. CE8.e

16.8.2.5.6. CE8.f

16.8.2.5.7. CE8.g

16.8.2.6. RA9

16.8.2.6.1. CE9.e

16.8.2.6.2. CE9.f

16.8.2.6.3. CE9.g

17. **Seguridad y cacheado en Yii 2** #ev2 (due: 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

19.1.2.2.2. Crear pruebas en formato Cest

19.1.2.3. Fixtures

19.1.2.3.1. `./yii fixture/generate <nombre>`

19.1.2.3.2. `./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

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. Ejercicios

19.5.1. Escribir pruebas funcionales para Muéveme

19.5.2. Generar documentación para Muéveme

19.5.3. Usar fixtures para generar datos de prueba para Muéveme

19.5.4. Verificar en CodeClimate si hay archivos fuente marcados como de mala calidad

19.6. Meta

19.6.1. Objetivos de la unidad

19.6.2. Resultados de aprendizaje y criterios de evaluación asociados

19.6.2.1. RA3

19.6.2.1.1. CE3.g

19.6.2.2. RA4

19.6.2.2.1. CE4.g

19.6.2.3. RA5

19.6.2.3.1. CE5.h

19.6.2.4. RA6

19.6.2.4.1. CE6.h

19.6.2.5. RA9

19.6.2.5.1. CE9.e

19.6.2.5.2. CE9.g

20. **Computación en la nube** #ev2 (due: 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. Meta

20.9.1. Objetivos de la unidad

20.9.2. Resultados de aprendizaje y criterios de evaluación asociados

20.9.2.1. RA1

20.9.2.1.1. CE1.c

20.9.2.1.2. CE1.d

20.9.2.1.3. CE1.e

20.9.2.2. RA9

20.9.2.2.1. CE9.e

20.9.2.2.2. CE9.f

20.9.2.2.3. CE9.g

21. **Servicios web con REST en Yii 2** #ev2 #opcional (due: 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. Meta

21.10.1. Objetivos de la unidad

21.10.2. Resultados de aprendizaje y criterios de evaluación asociados

21.10.2.1. RA2

21.10.2.2. RA3

21.10.2.3. RA7

21.10.2.3.1. CE7.a

21.10.2.3.2. CE7.b

21.10.2.3.3. CE7.c

21.10.2.3.4. CE7.d

21.10.2.3.5. CE7.e

21.10.2.3.6. CE7.f

21.10.2.3.7. CE7.g

21.10.2.4. RA9

21.10.2.4.1. CE9.c

21.10.2.4.2. CE9.e

21.10.2.4.3. CE9.f

21.10.2.4.4. CE9.g

22. **Contenedores** #ev2 #opcional (due: 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

22.2.3.1. docker-compose.yml

22.3. PHPDocker

22.4. Meta

22.4.1. Objetivos de la unidad

22.4.2. Resultados de aprendizaje y criterios de evaluación asociados

22.4.2.1. RA1

22.4.2.1.1. CE1.c

22.4.2.1.2. CE1.d

22.4.2.1.3. CE1.e

22.4.2.2. RA9

22.4.2.2.1. CE9.e

22.4.2.2.2. CE9.f

22.4.2.2.3. CE9.g

6. Orientaciones metodológicas

Este módulo profesional contiene la formación necesaria para desempeñar la función de desarrollo de aplicaciones y servicios destinados a su ejecución por servidores en entornos Web.

La función de desarrollo de aplicaciones para servidores Web incluye aspectos como:

- La creación de aplicaciones de servidor que generan interfaces Web como resultado de su ejecución.
- La programación de métodos para almacenar, recuperar y gestionar mediante documentos Web información disponible en almacenes de datos.
- La generación de servicios reutilizables y accesibles mediante protocolos Web.
- El desarrollo de aplicaciones basadas en información y funcionalidades distribuidas.

Las actividades profesionales asociadas a esta función se aplican en el desarrollo y la adaptación de servicios y aplicaciones para servidores de aplicaciones y servidores web.

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

- El análisis de los métodos de generación dinámica de documentos Web.
- La integración del lenguaje de marcas con el código ejecutable en el servidor Web.
- El análisis, diferenciación y clasificación de las características y funcionalidades incorporadas en los entornos y lenguajes de programación de los servidores Web más difundidos.
- La utilización de características y funcionalidades específicas de los lenguajes de programación seleccionados.
- La modificación del código existente en soluciones Web heterogéneas para su adaptación a entornos específicos.
- El análisis y la utilización de funcionalidades aportadas por librerías generales y específicas de programación web en entorno servidor.
- La utilización de librerías para incorporar interactividad a los documentos Web generados de forma dinámica.

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 la tecnología web, separando el entorno cliente del servidor, seguido de la profundización en el desarrollo de aplicaciones web con lenguajes de script embebidos tanto directamente como

mediante el uso de un framework apropiado para tal fin, finalizando con una introducción a las nuevas tendencias en el despliegue de aplicaciones web desde la perspectiva del entorno servidor.

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

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

7. Recursos

7.1. 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.

7.2. 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.

7.3. Online

- Plataforma **Ágora**[1] para el seguimiento general del módulo, incluyendo distribución de material y entrega de ejercicios y exámenes.
- **GitHub** y **GitHub Classroom**[2] como herramienta centralizada 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**[3], que les ofrece servicios y herramientas con descuentos de hasta el 100 % con respecto al precio de mercado.

[1] `http://agora.iesdonana.org`
[2] `https://classroom.github.com`
[3] `https://education.github.com/pack`

- Dokuwiki[4] como herramienta de documentación colaborativa.

7.4. Bibliografía

- Apuntes y documentación *online*[5] suministrados por el profesor.
- Documentación de Git[6] y GitHub [7].
- Documentación de PHP[8].
- Documentación de Yii 2 Framework[9].
- Documentación de PostgreSQL[10].
- Complementaria: «Desarrollo web en entorno servidor». López Sanz, M. y otros. Editorial RA-MA.

8. Atención a la diversidad

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

9. 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

[4] `http://wiki.iesdonana.org`
[5] `https://dwese.iesdonana.org`
[6] `https://git-scm.com/book/en/v2`
[7] `https://help.github.com`
[8] `https://php.net`
[9] `https://www.yiiframework.com`
[10] `https://www.postgresql.org`

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.
- **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.

10. 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.

10.1. 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.*»[11]

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.

[11] `https://twitter.com/RAEinforma/status/1111565711653113856`