Proyecto integrado

Presentación de propuestas

1. Se abre en Ágora un apartado donde el alumno subirá su propuesta de Proyecto. La fecha tope es de treinta días de antelación a la fecha prevista para el inicio del proyecto. Si no se envía ninguna propuesta, se le adjudicará directamente un proyecto propuesto por el Departamento.
2. En el plazo de una semana, los profesores aportan sugerencias y comentarios de cara a que el alumno lleve a cabo las modificaciones correspondientes, si procede. De ser así, se le da un plazo de diez días para que pueda volver a presentar la propuesta ya corregida.
3. La propuesta también puede rechazarse directamente si el profesorado considera que no cumple los mínimos necesarios en cuanto a orientación, temática, calidad, funcionalidad, complejidad, adecuación, viabilidad, etc. En este caso, a discreción del profesorado, se puede optar por:
    - Dar una nueva oportunidad al alumno para que haga una nueva propuesta, o
    - Asignar al alumno directamente una propuesta (completa o semicompleta) de Proyecto elaborado por el profesorado.
4. La convocatoria se consumirá a menos que el alumno renuncie antes.

Contenido de la propuesta

1. Descripción general del Proyecto.
    - Nombre.
    - Funcionalidad principal de la aplicación.
    - Objetivos generales.
        - *A diferencia de los casos de uso, los objetivos no tienen principio ni fin. Por ejemplo:*
            - *Objetivo: "gestionar los alquileres y las devoluciones de las películas".*

        - *Casos de uso: "alquilar una película", "devolver una película".*
2. Elemento de innovación.
    - *Aspecto, función o tecnología novedosa o innovadora no tratada directamente en clase y que será necesario investigar para desarrollar adecuadamente el Proyecto.*
3. Catálogo de requisitos, con:
    - Código.
        - *Se codifican todos con **R1**, **R2**, …*
    - Descripción corta.
        - *Breve resumen del requisito, en una sola frase.*
    - Descripción larga.
        - *Descripción detallada, tan larga como sea necesario (varios párrafos).*
    - Prioridad:
        - Mínimo.
            - *Requisito exigido por los profesores con carácter general para todos los proyectos.*
        - Importante.
            - *Requisito fundamental para la aplicación, sin la cual ésta no cumpliría adecuadamente con el cometido para la que fue ideada.*
        - Opcional.
            - *Requisito que complementa la funcionalidad de la aplicación sin resultar absolutamente necesario.*
            - ***Los requisitos opcionales pueden variar en definición y número durante el desarrollo del Proyecto.** Los importantes, no.*
    - Tipo:
        - Funcional.
            - *Descrito desde el punto de vista del usuario (**casos de uso**).*
            - *Asociado a los objetivos generales.*
        - Técnico.
        - Información.
            - *Describe los requisitos de almacenamiento de información (qué datos hay que almacenar de cada entidad/modelo).*
    - Complejidad.
        - *Se estima la dificultad de implementar cada requisito en **fácil**, **media** o **difícil**.*
    - Entrega planificada.
        - *Se planifica la implementación de cada requisito funcional en la entrega **v1**, **v2** ó **v3**.*

Incluir una tabla como esta a modo de resumen en el catálogo de requisitos:

Requisito (cód. + desc. corta)	**Prioridad**	**Tipo**	**Complejidad**	**Entrega**
R1. Dar de alta una película	Importante	Funcional	Fácil	v1
R2. Alquilar una película	Importante	Funcional	Media	v2
...	...	...	...	...

Entrega final

Contenido de la guía general

0. Portada e índice de contenidos.
    a. *Incluyendo el nombre de la aplicación.*
1. Descripción general del Proyecto.
    a. Funcionalidad principal de la aplicación.
    b. Objetivos generales.
    c. URL del repositorio en GitHub.
    d. URL de la documentación en GitHub Pages.
2. Instrucciones de instalación y despliegue.
    a. En local.
    b. En la nube.
3. Catálogo de requisitos.
    a. Definición detallada.
        i. *Indicando código, descripción corta, descripción larga, prioridad, tipo, complejidad, entrega planificada, entrega realizada, nº de incidencia asociada en GitHub.*
    b. Cuadro resumen.
        i. *En una tabla, Indicando todo lo anterior menos la descripción larga.*
4. Manual básico de usuario.
    a. *Incluir capturas de pantalla.*
5. Decisiones adoptadas y su justificación.
6. Dificultades encontradas y soluciones aplicadas.
    a. *Incluir aquí el elemento de innovación.*
7. Diagramas.
    a. De clases.
    b. De estructura lógica de datos.
8. Glosario de términos.
9. Conclusiones.
10. Anexos.
    a. *Documentos específicos vinculados a determinados requisitos funcionales (“prueba del seis” y similares).*

Defensa (orientativo)

1. Preparación: 5 minutos.
2. Demostración de uso: 15 minutos.
3. Descripción técnica: 15 minutos.
4. Preguntas: 5 minutos.
5. Deliberación del tribunal: 5 minutos.

Preparación del código

- Usar el script `proyecto.sh` para crear un nuevo proyecto:
  ```
  $ cd ~/web
  $ proyecto.sh miproyecto
  ```
- Para trabajar correctamente con fechas, horas, instantes y zonas horarias:
  - Configurar PostgreSQL para trabajar con `timezone = 'UTC'` en `postgresql.conf` (por defecto vale `'localtime'`, tomando la del sistema). En Heroku está correcto ya.
  - Configurar PostgreSQL para trabajar con `intervalstyle = 'iso_8601'` en `postgresql.conf` (por defecto vale `'postgres'`). En Heroku tiene el mismo valor por defecto, por lo que tenemos que cambiarlo cada vez que establecemos una conexión a la base de datos. Para ello, añadimos en la configuración de la aplicación:
    ```
    'db' => [
        // ...
        'on afterOpen' => function ($event) {
            $event->sender->createCommand("SET intervalstyle = 'iso_8601'");
        }
    ]
    ```
  - Configurar PHP para trabajar con `date.timezone = 'UTC'` en `php.ini` (por defecto toma la zona horaria del sistema). En Heroku está correcto ya.
  - Configurar la aplicación Yii2 en `config/web.php` para que `'timeZone' => 'Europe/Madrid'`, o la zona horaria que interese.
  - Si cada usuario configura su propia zona horaria, se puede añadir lo siguiente en `config/web.php`:
    ```
    'on beforeRequest' => function ($event) {
        Yii::$app->setTimeZone(
            Yii::$app->user->isGuest ?
            'Europe/Madrid' :
            Yii::$app->user->identity->zona_horaria
        );
    }
    ```

Requisitos mínimos

Generales

1. Requisitos perfectamente definidos y convertidos en **incidencias** (*issues*) de GitHub.
2. Código fuente publicado en GitHub.
3. Estilo del código según las normas internas de Yii2 para el código y para las plantillas de las vistas.
4. Tres lanzamientos (*releases*) etiquetados en el repositorio como **v1**, **v2** y **v3**.
5. `README.md` en el directorio raíz con la descripción principal del proyecto.

6. Documentación generada con `yii2-apidoc` y publicada en GitHub Pages a partir del contenido del directorio `/docs`:
    a. Contenido:
        i. Guía general.
        ii. API.
    b. Formato: GitHub flavored Markdown (fuente) y HTML (resultado).
    a. Usar script `publicar_doc.sh` contenido en la raíz del proyecto.
    c. *Opcional*: conversión a PDF.
7. Administración y resolución de todas las incidencias notificadas en GitHub.
8. Usar etiquetas e hitos:
    a. Etiquetas: **`mínimo`**, **`importante`**, **`opcional`** (además de las ya existentes).
    b. Hitos: **`v1`**, **`v2`**, **`v3`** (con fechas de entrega aproximadas).
9. La rama `master` debe reflejar en todo momento el estado más estable de la aplicación, de manera que:
    a. La rama `master` no debe contener bugs conocidos.
    b. El desarrollo deberá hacerse en otras ramas creadas a tal efecto (una distinta por cada funcionalidad) y se irán combinado con la `master` una vez que se haya implementado la funcionalidad correspondiente.
    c. Cada rama debe ir asociada con una incidencia. El nombre de la rama debe empezar por el número de la incidencia correspondiente (p. ej. `17-login`).
    d. La release actual en Heroku corresponderá siempre con el último commit de la rama `master` (usar los *deploys automáticos* de Heroku conectando la aplicación de Heroku con la rama `master` de GitHub).
10. Usar Waffle para la gestión general del proyecto.
11. Al final de cada iteración:
    a. Se realiza el lanzamiento que toque (**`v1`**, **`v2`** o **`v3`**), etiquetando el commit correspondiente con el hito adecuado.
    b. Se actualiza y publica la documentación.
    c. Al final del Proyecto, se tiene que cumplir lo siguiente:
        i. Todas las incidencias cerradas con su debida justificación.
        ii. En el *backlog* sólo pueden quedar tarjetas con prioridad **`opcional`**.
        iii. El lanzamiento **`v3`** desplegado en la nube.
        iv. La documentación correctamente actualizada y publicada.

Desarrollo web en entorno cliente

1. **Validación** de los campos de los formularios.
2. **Gestión de ventanas** (Gestión de la apariencia de las ventanas. Creación de nuevas ventanas y comunicación entre ventanas).
3. Interactividad a través de **mecanismos de manejo de eventos** intuitivos y eficaces.
4. Uso y manipulación de las características del modelo de objetos del documento **(DOM).**
5. Uso de mecanismos de almacenamiento en el lado del cliente.
6. Uso de la librería **JQUERY.**
7. Incluir al menos un plugin no trabajado en clase.
8. Utilización de mecanismos de comunicación asíncrona: **AJAX.**

Desarrollo web en entorno servidor

Para los alumnos que hayan cursado el framework Yii2:

1. PHP 7.1.
2. Yii2 Framework versión 2.0.10 ó superior.
3. PostgreSQL versión 9.6 ó superior.
4. Despliegue de la aplicación en la plataforma Heroku.
5. Pruebas funcionales con Codeception.
6. Estilo y mantenibilidad del código fuente validados por Code Climate.
7. La aplicación ha de ser escalable.
8. La aplicación debe hacer en algún momento un uso apropiado de la tecnología AJAX.

Para los alumnos repetidores de cursos anteriores que hayan cursado el framework CodeIgniter en lugar de Yii2:

1. PHP 5.6.
2. CodeIgniter 3 ó superior.
3. PostgreSQL versión 9.3 ó superior.
4. La aplicación ha de ser escalable.
5. La aplicación debe hacer en algún momento un uso apropiado de la tecnología AJAX.

Diseño de interfaces web

1. Para estructurar el contenido se utilizarán las etiquetas semánticas de HTML5.
2. Todo lo relacionado con la presentación se trabajará mediante CSS.
3. El diseño será flexible.
4. Existirán transiciones, transformaciones, animaciones y contenido multimedia.
5. Uso de microdatos.
6. Se deberá comprobar que el código realizado supera:
    a. El validador para HTML5, CSS3.
    b. Nivel de accesibilidad AA.
    c. Prueba del seis.
7. Implementar el diseño para resoluciones grandes y pequeñas.
8. Comprobar que el diseño es correcto en los siguientes navegadores:
    a. Internet Explorer.
    b. Chrome.
    c. Mozilla Firefox.
    d. Opera.

Despliegue de aplicaciones web

1. Realizar el despliegue en un Host:

    - Utilizando algún servicio gratuito de hosting como los vistos en clase
    - Instalar / configurar o solicitar el software necesario para desplegar el proyecto.
2. Realizar un despliegue en un servidor local usando y configurando tres máquinas virtuales para:
    - Crear un servicio de Nombres de dominio.
    - Gestionar y administrar el servidor Apache tanto en Windows como Linux:
        - Instalar el servidor y configurarlo.
        - Configurar directivas.
        - Usar directorios virtuales y redireccionamientos.
        - Usar diferentes módulos estáticos y dinámicos.
        - Usar autenticaciones.
        - Usar ficheros de configuración personalizada de directorios.
        - Usar HTTPS y certificados Digitales.