PEER ASSESSMENT

CONCURSO DE CARTELES SOBRE LA FIESTA DE LA MERCÉ 2017

Pilar Bielsa Martínez

Isaac García Sánchez

2016-2017

2º - DAW

INSTITUT

Jaume Balmes

ÍNDICE:

INTRODUCCIÓN 3
1. Mapa de software implantado 4
2. Estudio de viabilidad. 5
3. ¿Qué es un Framework? 8
4. ¿Qué es el Peer Assessment? 14
5. Definición del proyecto. 18
    5.1. Objetivos 18
    5.2. Requisitos 19
6. Implementación 20
    6.1. Vistas 20
    6.2.Controladores 23
        6.2.1. Función de asignar proyectos 23
    6.3. Multiidioma. 23
7. Base de datos 24
Conclusión 25
Bibliografía 26

INTRODUCCIÓN

Lo que se pretende con este proyecto, es llevar a cabo un sistema de auto evaluación de concursos que hará más fácil la elección del producto final cuando haya mucho material a evaluar y un gran número de participantes. Los mismos participantes evaluaran sus mismos proyectos.

Se implementará en PHP. Será el lenguaje usado con mayor frecuencia durante el desarrollo del proyecto. Además, gracias a la utilización de un entorno de trabajo especifico, nos permitirá seguir un conjunto de conceptos, prácticas y criterios unificados que nos servirán de base para la organización y desarrollo de la aplicación. También se hará necesario el uso de una base de datos, en la cual se guardará toda la información relacionada con la app.

Las circunstancias por las cuales hemos decidido hacer este trabajo nacen de la idea supeditada por uno de los profesores. Ha sido él quien realmente nos ha dado la idea. Sí bien, nosotros hemos llevado a cabo de forma íntegra todo el diseño de la aplicación desde su inicio.

Los objetivos de este proyecto son poder acabarlo. Que sea funcional. Por un lado, que permita a los participantes registrarse y poder hacer login entrando así en su área. Por el otro, que se puedan subir los proyectos y al cabo de un tiempo, se abra la fase de evaluación, se repartan los proyectos de forma aleatoria entre los participantes y ellos mismos efectúen las evaluaciones. También sería bueno conseguir que tras el proceso de evaluación, se enviarán mails a los participantes con los resultados obtenidos así como notificación al ganador. Todo ello aderezado con un panel de gestión de administrador.

Respecto a la metodología usada, nos hemos basado en los trabajos de la rama tecnológica que son una particularidad de la científica. Esto comportará un desarrollo de un proceso tecnológico que sigue el método del proyecto. Todo nos llevará a una serie de fases tales como, determinar el problema, escoger la mejor solución, planificar, desarrollar y comprobar y evaluar.

1. Mapa de software implantado.

A continuación se define el software implantado para llevar a cabo el proyecto.

2. Estudio de viabilidad.

2.1. Introducción

Se plantea la necesidad de evaluar múltiples proyectos sin contar con un equipo de evaluadores definido como tal. Es por eso que se plantea la creación de una aplicación web que se encargue de asignar aleatoriamente los proyectos a los participantes y estos se evalúen entre ellos.

2.1.1. Declaración del problema

El problema presentado, nace de la necesidad de poder evaluar múltiples proyectos de forma simultánea. En este caso, evaluar carteles en base a unas condiciones estipuladas. Dicha evaluación ha de ser entre iguales, es decir que todos se evalúan entre ellos. Más adelante se explicará en qué consiste esta técnica.

2.1.2. Entorno de implementación

El entorno de implementación se limita a un par de equipos portátiles con 4GB de RAM como mínimo y 100GB des espacio libre en disco. Esto en cuanto al entorno físico.

El entorno software, se requiere usar la última distribución de XAMPP, con Apache y MySQL operativos, así como la última versión de PHP. Además es necesario instalar el entorno en cuestión, es decir Laravel, con Composer.

2.1.3. Restricciones

Se han hallado restricciones en cuanto a poder implementar el sitio web en un hosting real. Además, hubiera sido una buena práctica para tener el sitio accesible desde cualquier parte.

Restricciones menores, han aparecido a la hora de asignar los proyectos en los que ha resultado extremadamente difícil hallar un algoritmo referente a la combinatoria sin repetición. Finalmente se subsanó.

2.2. Resumen y recomendaciones de gestión

2.2.1. Hallazgos importantes

Se han hecho múltiples hallazgos en la materia. Sin ir más lejos en el desarrollo de proyectos con Laravel. Se han afianzado conocimientos en la materia y estos han dado sus frutos a lo largo de todo el desarrollo.

También se ha hallado la forma de dotar al sitio de una interfaz totalmente responsive, es decir que se adapte a los dispositivos móviles sin problemas y seguir siendo funcional a pesar de cambiar el modo de visualización.

2.2.2. Comentarios

No se perciben comentarios a destacar.

2.2.3. Recomendaciones

También se han hallado formas inteligentes a la hora de gestionar el proyecto, que is bien, no han sido implementadas en su totalidad, son herramientas poderosas que sirven para controlar las versiones, hablamos de GitHub. Se recomienda un uso más intenso de GitHub sobretodo por todos los miembros del proyecto para poder llevar un control más exhaustivo de las versiones.

2.2.4. Impacto

2.3. Alternativas

2.3.1. Configuraciones del sistema alternativas

2.3.2. Criterio utilizado en la selección del enfoque definitivo

2.4. Descripción del sistema

2.4.1. Declaración resumida del ámbito

2.4.2. Viabilidad de los elementos asignados

2.5. Análisis de coste-beneficio

Este proyecto, se estima presentarlo en un concurso público para atender requerimientos de las comisiones de fiestas de cada ámbito. Es decir se estima un beneficio estándar y único en el que obviamente los costes deben ser cubiertos para poder dar luz verde a la implantación del proyecto.

Puesto que se trata de un proyecto de evaluación entre iguales, su implementación está, también muy relacionada en el ámbito académico y se deberá negociar con los centros escolares las cuotas necesarias para implementar el sistema y adecuarlo a las necesidades solicitadas. Teniendo en cuenta el tiempo necesario y el coste sufrido de adecuar el aplicativo a las necesidades del cliente y poder tener así beneficios.

Por último, se estiman unos beneficios elevados, dependiendo así del sueldo de los miembros del proyecto así como del tiempo invertido y de las modificaciones que sean necesarias para adecuarlos a las necesidades.

Si se hace un análisis más normal en el que se tengan en cuenta únicamente los costes y los beneficios se obtiene lo siguiente:

Costes	Beneficios
Salario por horas de los desarrolladores	Sistema fiable de asignación de proyectos
Adquisición de licencias si fuera necesario	Ahorro de trabajo al no necesitar evaluadores externos.
Mantenimiento de las infraestructuras físicas tales como despacho y sistemas.	Se ahorra mucho al no depender de evaluadores ya que los proyectos se evalúan entre los participantes.
	Sistema autónomo, ya que puede funcionar sin tener que dar un mantenimiento muy a fondo.

2.6. Evaluación del riesgo técnico

2.7. Consideraciones legales.

Se mencionan las leyes a tener en cuenta para este tipo de proyecto:

- Ley Orgánica de Protección de Datos (LOPD): para proteger los datos que proporcionan los participantes.
- La Ley de Comercio Electrónico (Ley 34/2002, de 11 de julio) (Ley de Servicios de la Sociedad de la Información y de Comercio Electrónico o LSSI) obliga a identificar al titular de una web salvo en aquellas páginas web puramente personales.
- Además, hay una obligación general que afecta a todas las páginas web, y es la de proporcionar al usuario de la web una información detallada sobre algunos aspectos concretos que le sirven de garantía y seguridad jurídica.

2.8. Otros asuntos específicos del proyecto

3. ¿Qué es un Framework?

Una definición sencilla para *framework*, sería un conjunto estandarizado de conceptos y criterios para apuntar a un tipo de problema particular que sirve como referencia para resolverlo. En pocas palabras se trata de un entorno de trabajo.

En el ámbito que nos ocupa, es decir, en el software, un *framework* se define como una estructura conceptual asistida con determinados módulos concretos que pueden servir de base para la organización y desarrollo de una aplicación. Es un compendio de reglas a tener en cuenta para poder desarrollar un proyecto en base a esas reglas y tenerlo todo encapsulado siguiendo, en nuestro caso el modelo vista controlador.

Arquitectura

El framework utilizado se basa en el patrón MVC[1] y en base a este patrón podemos decir, de forma breve:

Modelo: Maneja las operaciones lógicas, y de manejo de información (previamente enviada por su ancestro), para resultar de una forma explicable. Cada miembro debe ser llamado, con su correcto nombre y en principio, con su verdadera naturaleza: el manejo de información, su complementación directa.

Vista: Expresa la última forma de los datos: la interfaz gráfica que interactúa con el usuario final del programa (GUI). Después de todo, a este miembro le toca evidenciar la información obtenida hasta hacerla llegar al controlador.

Controlador: Se controla el acceso a nuestra aplicación, y esto puede incluir: archivos, scripts, o programas, cualquier tipo de información que permita la interfaz. Así, podremos diversificar nuestro contenido de forma dinámica, y estática (a la vez) pues, solo debemos controlar ciertos aspectos.

A lo que respecta a la parte lógica de un *framework*, esta se basa en el MVC[2] y además necesita de un sistema encargado de rutear, es decir, dividir las peticiones sin tantos condicionantes.

[1] **MVC**: Modelo, vista controlador. Controlador→Modelo→Vista. Es importante tener este patrón claro a la hora de trabajar con un *framework*, sobre todo si este se basa en este patrón como base de trabajo. Hay que tener en cuenta que este modelo difiere un poco según quien lo defina y existen controversias sobre donde debería ir, a veces cada funcionalidad.

[2] **Controlador**: Ser capaz de manejar rutas, archivos, clases, métodos y funciones.
Modelo: es como un script habitual en el servidor, solo que agrupado bajo un modelo reutilizable.
Vista: como incluyendo cualquier archivo en nuestra ejecución, muy simple.

A continuación nos centramos, en las paginas sucesivas, a detallar el *framework* utilizado y el porqué de esta elección.

El framework escogido para este proyecto es Laravel, Laravel 5.4 para ser más exactos. El motivo es porque durante este curso hemos tratado con él y por tanto se ha decidido arriesgar a implementar un proyecto más grande con este *framework* como base.

Laravel 5.4. **Un poco de historia**.

Filosofía principal: Desarrollar código php de forma simple y elegante, evitando así el código espagueti[3].

Creado en 2011 y desarrollado por Taylor Otwell[4]. Laravel, intenta aprovechar los mejor de otros frameworks además de las características de las últimas versiones de PHP. La mayor parte de Laravel está formada por dependencias de Simfony, esto conlleva a la hora de desarrollar Laravel un desarrollo, también de sus dependencias.

Laravel 5.4. Características.

- Sistema de ruteo, también RESTful[5]
- Blade, Motor de plantillas
- Peticiones Fluent
- Eloquent ORM[6]
- Basado en Composer[7]
- Soporte para el caché
- Soporte para MVC
- Usa componentes de Symfony
- Adopta las especificaciones PSR-2 y PSR-4

Fig. 1.1. *Logo de Laravel*

Fig. 1.2. *Logo de Composer*

[3] **Código espagueti**: Clasificado como un anti-patrón de programación. Se define así a los programas que tienen una estructura compleja e incomprensible. El motivo de su denominación, es que este código se asemeja a un plato de espaguetis, un montón de hilos enmarañados y anulados.
[4] Creador de Laravel, Lumen, Forge, Envoyer, y Spark. Es desarrollador web.
[5] Se dice que una aplicación es RESTful, cuando sigue un conjunto de principios de arquitectura tales como: Un protocolo cliente/servidor sin estado, un conjunto de operaciones bien definidas que se aplican a todos los recursos de información, una sintaxis universal para identificar los recursos y el uso de hipermedios, tanto para la información de la aplicación como para las transiciones de estado de la aplicación: la representación de este estado en un sistema REST son típicamente HTML o XML.
[6] El elocuente ORM ofrece una implementación ActiveRecord simple para trabajar con su base de datos. Cada tabla de la base de datos tiene un "modelo" correspondiente que se utiliza para interactuar con esa tabla.
[7] Se trata de un sistema de gestión de paquetes para la programación en PHP. Nos provee de una forma estándar el manejo de dependencia de php y la librerías necesarias. Desarrollado por Nils Adermann i Jordi Boggiano. La primera versión salió en 2012.

Si bien ya se ha mencionado antes, vamos a detallar el funcionamiento de Laravel basado en el patron MVC, para tratar de ver las particularidades del framework en cuestión.

Hay que tener en cuenta, que en Laravel no se desarrolla siguiendo un estricto patrón MVC, se propone para ello usar *Routes with Closures*[8] con el objetivo de hacer el código más claro. No obstante, se puede usar el MVC tradicional.

Modelo en Laravel.

Como se mencionó con anterioridad, Laravel dispone de un sistema de mapeo de datos relacional llamado Eloquent ORM que facilita la creación de modelos. Se funda en el patrón active record[9]. Para crear modelos es tan sencillo como desde la consola del sistema, dentro de la carpeta del proyecto:

```
php artisan make:model Libro
```

Donde User es el nombre del modelo a crear. Esto genera un archivo php dentro de la carpeta app que es donde se almacenan todos los modelos.

```php
use Illuminate\Database\Eloquent\Model;

class Libro extends Model {

    protected $table = 'tb_libros';
}
```

Fig.1.3. *Modelo creado desde comando, recien creado no incluye variables, el aquí mostrado es solo de ejemplo.*

Realmente, el código anterior puede llegar a ser más simple aún. En el caso que el nombre de la tabla coincida con el nombre de la clase. Ya que Laravel, usa el paradigma de programación donde se favorece "la convención sobre la configuración". Y si ahora necesitamos disponer un listado, en una determinada ruta, por ejemplo en: http://mi-aplicacion.com/libro/listar, entonces, sólo haría falta crear la Ruta e interactuar con el modelo 'Libro' anteriormente creado, tal que así:

[8] Closure o clausura. Se trata de funciones evaluadas en un entorno que contiene varias variables dependientes de otro entorno. Cuando se llama al Closure, este puede acceder a estas variables. El uso explicito de Closures se asocia con la programación funcional. También se puede dar el caso de que una Closure aparezca cuando una función está definida dentro de otra función.

[9] **Patrón active record**: Es un patrón de arquitectura que se halla en aplicaciones que almacenan sus datos en Bases de datos relacionales. La interfaz de un cierto objeto debe incluir funciones como por ejemplo insertar (INSERT), actualizar (UPDATE) o eliminar (DELETE) y propiedades que correspondan de cierta manera a las columnas de la base de datos asociada.

```
Route::get('libro/listar', function() {
    $libros = Libro::all();
    return View::make('mi_vista', $libros);
});
```

Fig.1.4. *Detalle de ruta para mostrar todos los libro de la base de datos. Se devuelve la vista con todos los libros*

Vista en Laravel

Laravel se basa en un sistema de procesamiento de plantillas llamado Blade. Con este sistema, se consigue un código limpio dentro de las vistas. El sistema Blade permite una sintaxis reducida en su escritura. Además gestiona la memoria caché muy bien lo que hace vistas rápidas. Lo que realmente dota a Blade de gran ventaja es su sistema de manejar plantillas.

Ejemplo de vista, estas constan de dos archivos, por un lado la plantilla en sí:

```
<!DOCTYPE html>
<html lang="es">
    <head>
        <meta charset="UTF-8">
        <title>@yield('titulo')</title>
    </head>
    <body>
        @yield('navegacion')
    </body>
</html>
```

Fig. 1.5. *Plantilla web. El código @yield() identifica al método donde como parámetro se indica el nombre de la zona desplegar.*

Y luego, el código de la vista, donde se define la plantilla a usar y la información de las distintas zonas a desplegar:

```
@extends('template')

@section('titulo')

@stop
@section('navegacion')

@stop
```

Fig. 1.6*. En primer lugar se identifica la plantilla a utilizar y luego se definen las zonas*

Controlador en Laravel

Permiten organizar el código en clases sin tener que escribirlo todo en las rutas. Cabe notar, que todos los controladores se extienden del BaseController (controlador principal). Para crear controladores, igual que con los modelos, se puede usar un comando para ello:

```
php artisan make:controller NameController --resource
```

Donde NameController es el nombre del controlador y con el parámetro --resource, conseguimos que nos cree dentro del archivo la estructura de funciones tales como índice, store, etc.

Como ejemplo de controlador:

```
class UserController extends BaseController {
    public function mostrarPerfil($id)
    {
        $user = User::find($id);;
        return View::make('user.profile', array('user' => $user));
    }
}
```

Fig. 1.7. *Detalle de controlador. Dentro de este se pueden añadir tantas funciones como sean necesario. En este caso se devuelve a la vista el perfil solicitado del usuario en función del id suministrado.*

Los controladores se pueden llamar desde las rutas de la siguiente forma:

```
Route::get('user/{id}', 'UserController@mostrarPerfil');
```

Fig.1.8. *Detalle de llamada al controlador a través de una ruta. Mediante petición get, se llamada la vista con un id definido y se llama al controlador necesario y al método en cuestión*

Migraciones en Laravel

Por último, para no hacer muy extenso este capítulo y mencionar todo lo relacionado con laravel, sí que haremos especial mención a las migraciones, útiles para confeccionar la base de datos del proyecto que nos ocupa.

Las migraciones son una excelente opción para controlar la base de datos, permitiéndonos una fácil modificación de la misma. Las migraciones están relacionadas con el schema builder de Laravel para una fácil construcción del esquema de la base de datos.

Para llevar a cabo una migración, usamos el siguiente comando desde la terminal abierta en el proyecto:

```
php artisan[10] make:migration create_aspectos_table
```

Lo que se consigue con esto es crear el archivo con el nombre especificado después de migration. No es más que una clase heredera de Migration en la que se define el esquema de la tabla a crear.

La nueva migración aparecerá en el directorio database/migrations. El fichero de migración contiene en el nombre la fecha la cual permite a Laravel determinar el orden de las migraciones.

Si nos centramos ahora en la estructura del fichero obtenido tras hacer el make:migration, observamos que la clase contiene dos métodos. El método up(), sirve para añadir nuevas tablas, columnas o índices a la base de datos, mientras que el método down(), sirve para llevar a cabo una especie de rollback sobre las operaciones ejecutadas por up ().

Ejemplo de función up():

```php
public function up()
{
    Schema::create('flights', function (Blueprint $table) {
        $table->increments('id');
        $table->string('name');
        $table->string('airline');
        $table->timestamps();
    });
}
```

Fig.1.9. *Detalle de función up para crear la tabla flight. Se añaden todas las columnas y el tipo de datos*

[10] Tanto en esta como en las instrucciones anteriores se usan comandos de artisan que es la línea de comandos propia de Laravel. Hay una gran variedad de instrucciones y estas se pueden consultar a través de: `php artisan list`.

4. ¿Qué es el Peer Assessment?

Las traducciones que encontramos en nuestro idioma para referirnos al Peer Assessment, podrían ser algo así como evaluación entre iguales, coevaluación (autoevaluación) o evaluación de pares. Teniendo en cuenta el significado, podemos proseguir con la explicación de en qué consiste el Peer Assessment.

Este proceso está relacionado con el aprendizaje y aún que es aplicable en el mundo de la enseñanza, en la relación alumno-profesor, también se puede usar en otros ámbitos como es el caso de este proyecto. Los participantes se evalúan así mismos y de esta manera no es necesario definir una comisión que se encargue de evaluar de forma individual todos los proyectos.

Se puede usar para favorecer el desarrollo del aprendizaje. Si bien, se presentan dificultades a la hora de llevar a la práctica esta modalidad de evaluación, entre ellas están las de tipo conceptual, institucional, relacional o aspectos referidos a la fiabilidad y validez de la evaluación entre iguales.

A pesar de las dificultades mencionadas con anterioridad, existen beneficios que pueden alcanzarse al hacer uso de este tipo de evaluación, como la mejora de los procesos y productos del aprendizaje, el desarrollo de estrategias interpersonales, la mejora de la capacidad para emitir juicios o el desarrollo de determinadas competencias académicas y profesionales. Otro beneficio que encontramos es el uso de varias evaluaciones que se pueden consensuar en grupo, en lugar de calificaciones individuales.

Tal y como Falchikov y Goldfinch[11] sugieren en un estudio: La evaluación entre iguales se puede utilizar con éxito en cualquier disciplina, área y nivel.

Pasos para implementar la evaluación entre iguales.

1. Establecer los criterios de la evaluación

Consiste en otorgar el estándar o modelo con que comprar el trabajo evaluado. Los criterios permiten focalizar la evaluación y hacer que todos los trabajos sean evaluados bajo las mismas reglas.

[11] **Nancy Falchikov y Judy Goldfinch**: Pioneros del método Peer Assessment, entre otros. Relacionados con la implantación de la evaluación entre iguales.
Falchikov (2001) conceptualiza la evaluación a pares como la evaluación que los estudiantes realizan del trabajo o logros de sus compañeros utilizando para ello criterios relevantes.

2. Definir las reglas de la coevaluación

La coevaluación implica el análisis minucioso del trabajo de otro participante. Esto puede implicar que el participante evaluado se sienta vulnerable o criticado cuando su proyecto se considera por debajo de los estándares esperados. Por eso es importante establecer algunas reglas básicas que regulen la forma en que la retroalimentación se da para generar confianza. Si mencionamos algún ejemplo para definir las reglas:

- Por cada comentario negativo, se ofrece un comentario positivo
- Se evitará el lenguaje discriminatorio.
- Se hará referencia al trabajo, no a la persona y por tanto se puede definir como anónimo o el uso de pseudónimos en el caso del proyecto que nos ocupa

3. El proceso de la coevaluación

La siguiente pauta ayudar a que la coevaluación sea constructiva y no destructiva.

- Escucha, mira, responde: No hay nada más perjudicial que sentirse ignorado o parcialmente escuchado cuando uno presenta su trabajo. Por eso ten en cuenta que es necesario respetar las contribuciones de los demás así como responder a las preguntas y sugerencias. En este caso es importante realizar bien la evaluación.
- Ser constructivo: Si gusta o no un proyecto, siempre debemos proporcionar críticas constructivas, proporcionando sugerencias de mejoría. En este caso dejando comentarios en la evaluación del proyecto.
- Mantén el foco: Concentra tus comentarios evitando divagar ya que esto va a ser difícil de interpretar y no será de mucha utilidad. Los comentarios detallados son más útiles para ayudar a tus compañeros a mejorar su trabajo.

4. La entrega de retroalimentación

Existen varias reglas a tener en cuenta cuando se entrega retroalimentación del trabajo de otro. Esto es especialmente importante si se ha pedido comentar algo que todavía está en desarrollo.

- Mantén una actitud positiva.
- Se consciente de la fase en la que se encuentra el proyecto. Si estás evaluando una etapa inicial, no critiques como si se tratara del producto final.
- Evita centrarte en cuestiones de menor importancia, a menos que sean las únicas cosas que están mal.

- Recuerda que todo el mundo tiene una manera diferente de hacer las cosas
- Se flexible y evita centrarte en cómo lo habrías hecho tú.
- Inicia y termina con algo positivo.
- Busca ser útil, no dañino.

Los pasos mencionados con anterioridad, están relacionados con la implantación del proceso de evaluación entre iguales aún así son perfectamente extrapolables al proyecto que os ocupa. Es decir, que muchos de los puntos son aplicables al sistema de evaluación de proyectos por parte de los participantes.

La evaluación entre iguales puede entenderse como una forma específica de aprendizaje colaborativo en el que los participantes, en este caso, realizan una valoración sobre el proyecto de todos los participantes.

Es interesante saber, que la evaluación entre iguales se puede dividir en tres categorías básicas, estas son:

- **Evaluación intra-grupo**: Basada en la evaluación dentro de los grupos de trabajo. Cada participante o grupo de estos valora el trabajo realizado por los otros participantes de forma individual durante un proyecto común, por ejemplo la presentación de carteles para el concurso de las fiestas populares.
- **Evaluación inter-grupo**: Evaluación realizada entre grupos. Se valora el trabajo realizado por los distintos grupos
- **Evaluación individual**: Los participantes evalúan el proyecto del aprendizaje individual de sus iguales.

Podríamos decir, por tanto, que nuestro proyecto se basa en la modalidad de evaluación intra-grupo, aun que también podría encajar mejor en la modalidad de evaluación individual ya que las evaluaciones se llevan a cabo de forma individual. aún que si fuera necesario el administrador o los administradores del sitio podrían ojear algunas evaluaciones en concreto.

Uno de los aspectos más importantes a tener en cuenta, a la hora de implementar un sistema de evaluación entre iguales, y además es objeto de preocupación es la Fiabilidad y la Validez.

En este contexto, la fiabilidad se define como el grado de coincidencias existentes en las evaluaciones realizadas por los distintos participantes sobre un proceso, en este caso sobre los proyectos. Y la validez como el nivel de similitud respecto a la evaluación del participante con la evaluación modelo, por así decirlo.

Como conclusión, esperamos que haya quedado claro en qué consiste la evaluación entre iguales. Esta puede ir acompañada de una base de ayuda o un rubrica[12].

A continuación se exponen de forma esquemática las ventajas e inconvenientes de usar la evaluación a pares, claro que estas están basadas en su uso en un ambiente académico:

Ventajas:

+ Mayor objetividad.

+ Permite ahorrar tiempo y que este sea más efectivo ya que son los propios.

+ Incrementa el aprendizaje y el rendimiento.

+ Estimula el pensamiento y el aprendizaje profundo y crítico

Desventajas:

- Se pueden ocasionar malos entendidos.

- Puede que los resultados no sean muy fiables.

[12] **Rubrica**: Conjunto de criterios y estándares, relacionados con objetivos de aprendizaje, evaluar un nivel de desempeño o una tarea. Se trata de una herramienta de calificación utilizada para realizar evaluaciones objetivas, un conjunto de criterios y estándares ligados a los objetivos de aprendizaje usados para evaluar la actuación de alumnos en la creación de artículos, proyectos, ensayos y otras tareas. Las rúbricas permiten estandarizar la evaluación de acuerdo con criterios específicos, haciendo la calificación más simple y transparente.

5. Definición del proyecto.

A continuación se detallan de forma más precisa los objetivos del proyecto. Así como los requisitos, tanto de software como de hardware.

5.1. Objetivos

Se detallan en forma de lista todas las funcionalidades que se pretenden consolidar en la aplicación:

- Entorno web totalmente responsivo y funcional.
- Permitir a los participantes que se puedan registrar mediante formulario de registro.
- Tras el registro, los participantes se podrán logar entrando en su área particular.
- Dotar al sitio con un Área del participante totalmente funcional desde donde se pueda, definir pseudónimo, gestionar la acción de subir proyecto o evaluarlos y poder consultar las condiciones de evaluación.
- Formulario que permita subir el proyecto a cada participante, permitir la subida de imágenes y que se guarden en la base de datos.
- Panel de administrador, accesible desde login diferente, solo para administradores.
- Dotar de control al panel de administrador, este deberá poder, añadir nuevos aspectos de evaluación. Divisar todas las campañas y saber cuál es la que se encuentra activa. Además de visualizar a todos los participantes y todos los proyectos.
- Estaría bien que el administrador pudiera gestionar a los participantes de alguna manera.
- Control sobre la asignación de proyectos, mediante botón.
- Implementar sistema de asignación de proyectos de forma aleatoria entre todos los participantes y sin que haya repeticiones y sin que se queden proyectos sin asignar.
- Tras asignación correcta de proyectos pasar a la fase de evaluación.
- Control de fases mediante fechas.
- Mostrar a los participantes, previa fase de evaluación activada, la lista de proyectos a evaluar.
- Guardar las evaluaciones en la base de datos y poder tratar esos datos.
- Tras concluir el período de evaluación, revisar todas las evaluaciones y obtener cuales han sido las mejores.
- Idear sistema de envío de mail a todos los participantes para que estos conozcan cuales han sido los resultados.
- Enviar mails personalizados a los que han obtenido una mejor evaluación y por tanto notificar al ganador.
- Dotar a la aplicación de un sistema multi-idioma completamente funcional y que funcione en todo el site-map de la aplicación.
- Buen tratado de los datos a partir de la base de datos, para ello, deben crearse relaciones entre tablas solidas y coherentes.
- Dejar reflejado en todo momento siembre el nombre del participante registrado.

5.2. Requisitos

A nivel de hardware, estos serían los requisitos mínimos para poder desarrollar el proyecto:

- Memoria RAM 4GB.
- Intel Pentium i5 2,5Ghz
- Disco duro con 100GB de capacidad mínima. SSD mejor, para que todo vaya más rápido.
- Estudio de trabajo dotado con dos pantallas para un trabajo más ágil.
- Acceso a la nube para almacenar documentos y poder compartirlos con otro miembros. Uso de GitHub y Google Drive.
- En última instancia, el uso de terminales móviles para ir probando la aplicación en un móvil real y poder comprobar su funcionamiento cuando se han incorporado muchas funciones. Es evidente que esto se puede probar solo con el navegador.

Con respecto al software, ya se ha dejado entrever un poco en apartados anteriores, de todas maneras a continuación se lista y se explican con más detalles los no mencionados con anterioridad:

- Sublime Text.
- PHPMyadmin.
- GIMP 2.8. Muy útil para el retocado grafico de imágenes. Sobre todo para la reducción de las mismas y dotarlas de transparencia en algunos casos.
- Navegador web. Imprescindible para poder ver todo el desarrollo. Gracias a los navegadores de hoy en día, nos permiten ver el diseño como si se tratase de un móvil y no hace falta usar uno físicamente.
- XAMPP. Necesario para trabajar con un servidor apache y la base de datos MySQL.

6. Implementación

6.1. Vistas

En esta parte se explicará todo lo relacionado con las vistas de la aplicación así como el diseño de las mismas.

Se divide el punto en función de las vistas creadas y se comentan aspectos a tener en cuenta en cada una de ellas.

El diseño de las vistas se ha hecho mediante el uso de Bootstrap, más concretamente del Bootstrap js Modal. Sí bien, laravel ya dispone de uno, se optó por usar uno propio para tener mayor flexibilidad a la hora de llevar a cabo diseños propios y desde 0.

Principal

Área Participante

Formulario Entrega Proyectos.

Evaluación de Proyecto.

A continuación, se muestra el código a tener en cuenta:

```
<table class="table table-hover text-centered container-fluid" align="center">
      <tr>
        <th></th>
        <th>{{ trans('languaje.evaluacionproyecto1') }}</th>
        <th>{{ trans('languaje.evaluacionproyecto2') }}</th>
        <th>{{ trans('languaje.evaluacionproyecto3') }}</th>
        <th>Evaluar</th>
      </tr>
     @foreach ($proyectos as $proyecto)
     <tr>
       <td>Proyecto {{ $proyecto->id }}</td>
       <td>{{ $proyecto->titol }}</td>
       <td><button class="buttonmodal"[13] type="button" data-toggle="modal"[14] data-target="#{{ $proyecto->id }}"[15]><img class="img-responsive" src="cartel/{{ $proyecto->id }}" alt="{{ $proyecto->titol}} " title="{{ $proyecto->titol }}"></button></td>
       <td>{{ $proyecto->descripcion }}</td>
       <?php $datosproyecto = $proyecto->id; ?>

      <td><a class="btn btn-primary" href="/evaluacion?<?php echo $datosproyecto; ?>">Evaluar</a></td>
```

[13] Class = "Buttonmodal": Necesario para dotar al sitio web de ventanas emergentes.
[14] data-toogle= "modal": También necesario para que los modals funcionen.
[15] data-target = "# nombremodal ": Es el nombre del modal. Es decir con esto se llama al modal creado más abajo. En este caso se define el id del proyecto y así los modals serán todos diferentes. Para su uso en etiquetas <a> omitir por href = "#nombremodal"

```
    </tr>
    @endforeach
</table>
```

Lo que se muestra, es una tabla con todos los proyectos asignados a un participantes para posteriormente ser evaluados. Mediante el foreach, se recorren todos los proyectos.

Se hace necesario idear una forma para poder pasar el id del proyecto a la evaluación posterior y que luego esta quede asociada al proyecto en cuestión.

También es preciso, a la hora de mostrar la lista de los proyectos a evaluar que esta se rellene solo con los proyectos asignados a ese participante. Esto se tiene en cuenta mediante el *id_participant*.

Además, al hacer click en las imágenes estas se abren a tamaño real para poder verlas más grande, esto se consigue gracias al uso de modals, funcionalidad integrada en Bootstrap:

```
@foreach($proyectos as $key => $proyecto)
    <div class="modal fade" id="{{ $proyecto->id }}" tabindex="-1"
role="dialog" aria-hidden="true">
      <div class="modal-dialog" role="document">
        <div class="modal-content">
            <img src="cartel/{{ $proyecto->id }}" alt="Proyecto1">
        </div>
      </div>
    </div>
@endforeach
```

Muestras reales de lo anterior:

Fig. 6.x. *Detalle final de la lista de los proyectos a evaluar, recogida de la base de datos.*

Evaluación.

Se muestra el detalle para construir la funcionalidad que permite evaluar a los proyectos en función de unos aspectos definidos:

```
<table class="table table-striped">
      <tr class="info">
            <th>{{ trans('languaje.evaluacion2') }}</th>
            <th>0</th>
            <th>1</th>
            <th>2</th>
            <th>3</th>
            <th>4</th>
            <th>5</th>
      </tr>
      @foreach($aspectos as $aspecto)
      <tr>
            <td>{{ $aspecto->descripcion }}</td>
            <td id="id0">
            <input    type="radio"    name="eval{{    $aspecto->id    }}"
id="eval{{ $aspecto->id }}-option0" value="0" autocomplete="off">
            </td>
            <td id="id1">
            <input    type="radio"    name="eval{{    $aspecto->id    }}"
id="eval{{ $aspecto->id }}-option1" value="1" autocomplete="off">
            </td>
            <td id="id2">
            <input    type="radio"    name="eval{{    $aspecto->id    }}"
id="eval{{ $aspecto->id }}-option2" value="2" autocomplete="off">
            </td>
            <td id="id3">
            <input    type="radio"    name="eval{{    $aspecto->id    }}"
id="eval{{ $aspecto->id }}-option3" value="3" autocomplete="off">
            </td>
            <td id="id4">
            <input    type="radio"    name="eval{{    $aspecto->id    }}"
id="eval{{ $aspecto->id }}-option4" value="4" autocomplete="off">
            </td>
            <td id="id5">
            <input    type="radio"    name="eval{{    $aspecto->id    }}"
id="eval{{ $aspecto->id }}-option5" value="5" autocomplete="off">
            </td>
      </tr>
      @endforeach
</table>
```

Se trata de una tabla, que en función del número de aspectos que haya en la base de datos. A medida que estos vayan creciendo, irá aumentando también la tabla ya que se crea una fila por cada aspecto a evaluar. Se define el name="" y el id="" de cada input con el id del aspecto así a la hora de evaluar se pasa la nota de cada aspecto en función del id y no hay ningún tipo de desajuste en los datos.

Muestra real de lo anterior:

Fig. 6.x. *Detalle final de la funcionalidad de evaluación del proyecto*

Includes

- Acciones

- Header

Panel Administrador.

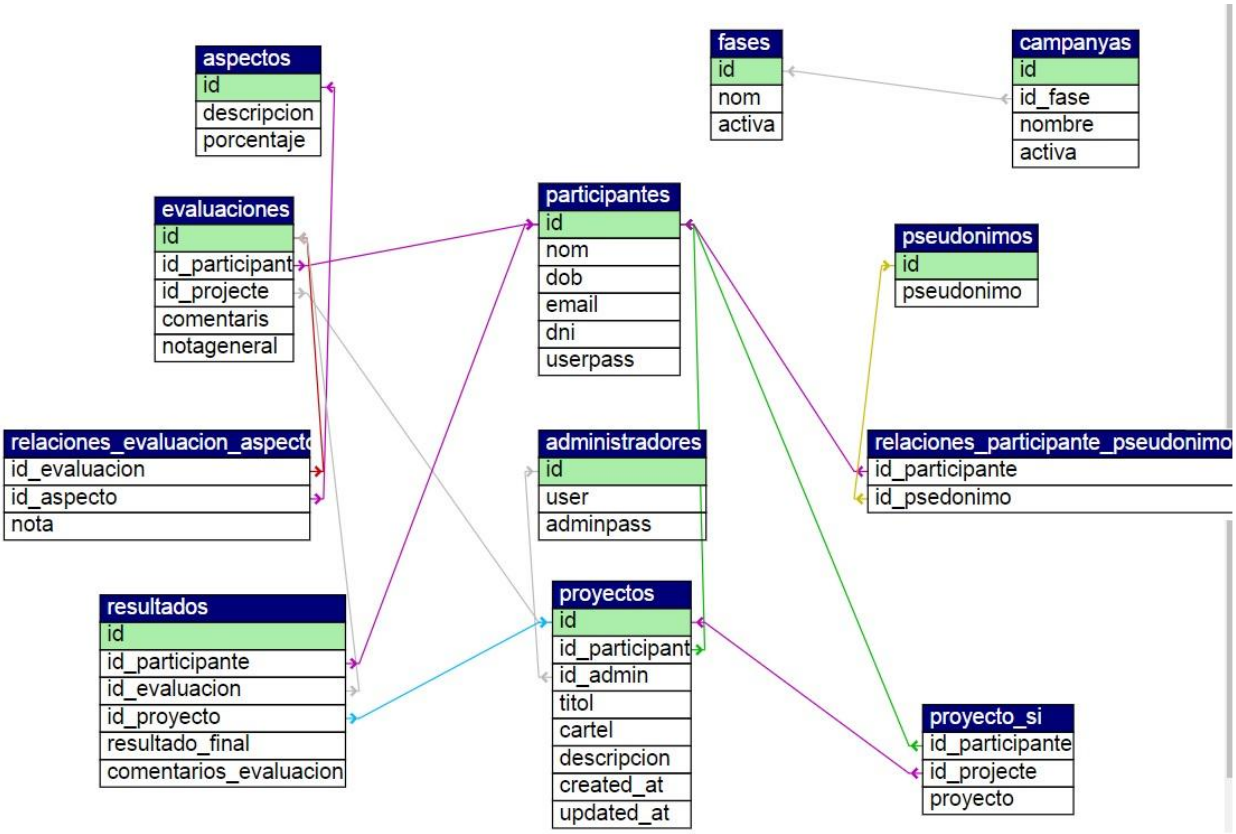
6.2.Controladores

6.2.1. Función de asignar proyectos

6.3. Multiidioma.

7. Base de datos

Esquema principal de la base de datos implementada en este proyecto:



Se presenta en el esquema anterior una idea de base de datos, que además es la implementada en el desarrollo final del proyecto, para tener controlada toda la información relacionada con los participantes, los proyectos así como las evaluaciones.

Conclusión

Bibliografía

JEFF. (2016). Aplicación multidioma en Laravel. Extraído de: <https://styde.net/aplicacion-multilenguaje-en-laravel-5-1/>

CLEMIR RONDÓN. (2016). Como usar el componente de traducciones en Laravel. Extraído de: <https://styde.net/como-usar-el-componente-de-traducciones-de-laravel/>

Laravel 5 Class 'Collective\Html\HtmlServiceProvider' not found. (2015). Extraído de: <https://stackoverflow.com/questions/32795154/laravel-5-class-collective-html-htmlserviceprovider-not-found-on-aws>

OLIVER VOGEL. (2015). Intervention Image. Extraído de: <http://image.intervention.io/getting_started/installation>

Using database to store images in Laravel 5.1. (2016). Extraído de: <http://www.core45.com/using-database-to-store-images-in-laravel-5-1/>

QAMAR UZMAN. (2014). Laravel / Intervention Image Class - class not found . Extraído de: <https://stackoverflow.com/questions/27374613/laravel-intervention-image-class-class-not-found>

ARLIND. (2015). Class 'Illuminate\Html\HtmlServiceProvider' not found Laravel 5. Extraído de: <https://stackoverflow.com/questions/28541051/class-illuminate-html-htmlserviceprovider-not-found-laravel-5>

SHALINI. (2016). Adding form action in html in laravel. Extraído de: <https://stackoverflow.com/questions/28984369/adding-form-action-in-html-in-laravel>

W. JASON GILMORE. (2015). Processing File Uploads With Laravel 5. Extraído de: <http://www.easylaravelbook.com/blog/2015/04/08/processing-file-uploads-with-laravel-5/>

Evaluación de Aula. Cómo implementar la coevaluación o evaluación de pares. Extraído de: <http://ww2.educarchile.cl/Portal.Base/Web/VerContenido.aspx?GUID=3f3203a8-0615-435d-8130-1ea019104c54&ID=224272>

VV.AA. (2012). La evaluación entre iguales: beneficios y estrategias para su práctica en la universidad. Extraído de: <http://www.revistaeducacion.mec.es/doi/359_092.pdf>

CRIFACACIAS. (2016). Beneficios de la evaluación entre pares. Extraidó de: <http://mooc.crifacacias.es/2016/11/11/beneficios-de-la-evaluacion-entre-pares/>

Ventajas y desventajas de la evaluacion entre pares y autoevaluacion. Extraído de: <https://yaninaarias.wikispaces.com/Ventajas+y+desventajas+de+la+evaluacion+entre+pares+y+autoevaluacion>

Peer assessment. <https://en.wikipedia.org/wiki/Peer_assessment>

Rúbrica. <https://es.wikipedia.org/wiki/R%C3%BAbrica_(docencia)>

Active record. <https://es.wikipedia.org/wiki/Active_record>

Composer. <https://en.wikipedia.org/wiki/Composer_(software)>

Sistema de gestión de paquetes. <https://es.wikipedia.org/wiki/Sistema_de_gesti%C3%B3n_de_paquetes>

Laravel. <https://es.wikipedia.org/wiki/Laravel>

Eloquent ORM. <https://laravel.com/docs/5.0/eloquent#introduction>

Database: Migrations. <https://laravel.com/docs/5.4/migrations>

THINKJAY. (2015). Access my Laravel site on another computer on the same network. Extraído de: <https://laracasts.com/discuss/channels/general-discussion/access-my-laravel-site-on-another-computer-on-the-same-network>