KITMAKER

Prueba LAMP Senior Developer

06 / 11 / 2014

Joan Gabriel Florit Gomila

joan.g.florit@gmail.com

@Joan_flo

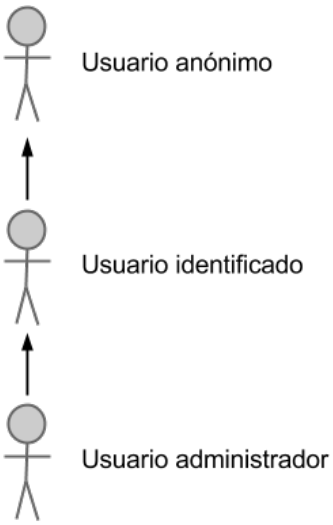
ÍNDICE

1 Análisis y planificación

1.1 Descripción general del sistema

La actual prueba consiste en crear un sencillo CMS en el que usuarios puedan dar de alta posts que luego sean visualizados por los demás. Existen unos usuarios administradores con potestad para censurar los posts inapropiados. Además, el sistema debe detectar desde qué móvil se está accediendo al blog.

1.2 Catálogo de usuarios



Como puede verse en el diagrama UML de la izquierda, la aplicación es usada por 3 tipos de usuarios:

- **Usuario anónimo.** Usuario sin identificar. Puede ver los posts públicos.
- **Usuario identificado.** Usuario que se ha identificado. Puede ver los posts públicos y los posts creados por él (aunque no sean públicos).
- **Usuario administrador.** Usuario identificado en el sistema y con el rol admin. Puede ver cualquier post y cambiar su estado.

1.3 Especificación de los requisitos

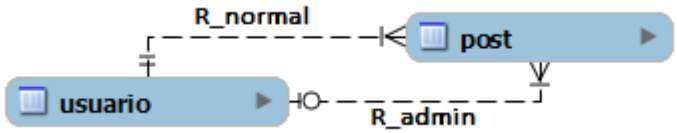
- Un usuario anónimo consulta los posts.
- Un usuario anónimo consulta la marca y modelo del móvil con el que navega.
- Un usuario anónimo se identifica en el sistema.
- Un usuario identificado crea un post.
- Un usuario administrador modifica el estado de un post.

2 Diseño

2.1 Base de datos

Se ha usado MySQL Workbench para diseñar la base de datos y generar los scripts.

Modelo Entidad-Relación



Existen dos relaciones entre ambas entidades:

- **R_normal.** Un usuario puede tener muchos posts (0..n); y un post pertenece a un único usuario (1..1).
- **R_admin.** Un admin puede haber aprobado/censurado muchos posts (0..n); y un post puede no haber sido aprobado todavía o ya haber sido aprobado/censurado (0..1).

Se puede dar el caso particular de que el creador del post sea a su vez el administrador.

Diccionario de datos

Leyenda	
Código	Significado
PK	Primary Key
FK	Foreign Key
NN	Not Null
AI	Auto Incremental
UQ	Unique Index

Post			
Campo	Tipo	Modificadores	Comentario
id_post	INT	PK, NN, AI	Identificador del post.
titulo	VARCHAR(100)	NN	Título del post.
contenido	VARCHAR(5000)	NN	Contenido textual del post.
create_time	TIMESTAMP	NN	Fecha de creación.
status	VARCHAR(1)	NN	Estado en el que se encuentra el post: - 'b' → borradores (valor por defecto). - 'p' → público. - 'c' → censurado.
email_creador	VARCHAR(255)	FK, NN	Usuario que ha creado el post.
email_admin	VARCHAR(255)	FK	Último administrador en aprobar o censurar el post.

Usuario			
Campo	Tipo	Modificadores	Comentario
email	VARCHAR(255)	PK, NN	Email del usuario normal o administrador.
username	VARCHAR(16)	NN, UQ	Nombre de usuario público.
password	VARCHAR(32)	NN	Contraseña encriptada usando MD5.
create_time	TIMESTAMP	NN	Fecha de registro del usuario.
status	VARCHAR(1)	NN	- 'n' → usuario normal (valor por defecto). - 'a' → admin.

Carga inicial de datos

Post						
id_post	titulo	contenido	create_time	status	email_creador	email_admin
1	Lorem ipsum 1	Lorem ipsum dolor sit amet, consectetur adipiscing elit...	31/10/14	p	joan.g.florit@gmail.com	joan.g.florit@gmail.com
2	Lorem ipsum 2	Donec ut elit sit amet nibh facilisis gravida. Vestibulum pretium augue eu tortor luctus placerat...	03/11/14	p	cgonzalez@kitmaker.com	joan.g.florit@gmail.com
3	Lorem ipsum 3	Sed nibh dui, varius in placerat vel, tincidunt sit amet ex...	31/10/14	p	xcunill@kitmaker.com	joan.g.florit@gmail.com
4	Lorem ipsum 4	Quisque eu nisl nec enim porta tempor et in neque...	31/10/14	b	cgonzalez@kitmaker.com	
5	Lorem ipsum 5	Sed faucibus urna metus, nec congue elit aliquam eu...	31/10/14	c	xcunill@kitmaker.com	joan.g.florit@gmail.com

Usuario				
email	username	password	create_time	status
joan.g.florit@gmail.com	joanflo	MD5('password1')	31/10/14	a
cgonzalez@kitmaker.com	cgonzalez	MD5('password2')	31/10/14	n
xcunill@kitmaker.com	xcunill	MD5('password3')	31/10/14	n

Scripts

- *Creación de la base de datos.*

```
-- -----------------------------------------------------
-- Schema jflorit_prueba_blog
-- -----------------------------------------------------
DROP SCHEMA IF EXISTS `jflorit_prueba_blog` ;
CREATE SCHEMA IF NOT EXISTS `jflorit_prueba_blog` DEFAULT CHARACTER SET utf8 COLLATE
utf8_general_ci ;
USE `jflorit_prueba_blog` ;
```

- *Creación de las tablas.* Los índices que crea por defecto MySQL (primary keys, foreign keys, unique index) son suficientes para el caso que nos ocupa.

```
-- -----------------------------------------------------
-- Table `jflorit_prueba_blog`.`usuario`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `jflorit_prueba_blog`.`usuario` ;

CREATE TABLE IF NOT EXISTS `jflorit_prueba_blog`.`usuario` (
  `email` VARCHAR(255) NOT NULL,
  `username` VARCHAR(16) NOT NULL,
  `password` VARCHAR(32) NOT NULL,
  `create_time` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `status` VARCHAR(1) NOT NULL DEFAULT 'n',
  PRIMARY KEY (`email`),
  UNIQUE INDEX `username_UNIQUE` (`username` ASC));

-- -----------------------------------------------------
-- Table `jflorit_prueba_blog`.`post`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `jflorit_prueba_blog`.`post` ;

CREATE TABLE IF NOT EXISTS `jflorit_prueba_blog`.`post` (
  `id_post` INT NOT NULL AUTO_INCREMENT,
  `titulo` VARCHAR(100) NOT NULL,
  `contenido` VARCHAR(5000) NOT NULL,
  `create_time` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `status` VARCHAR(1) NOT NULL DEFAULT 'b',
  `email_creador` VARCHAR(255) NOT NULL,
  `email_admin` VARCHAR(255) NULL,
  PRIMARY KEY (`id_post`),
  INDEX `fk_post_usuario_creador_idx` (`email_creador` ASC),
  INDEX `fk_post_usuario_admin_idx` (`email_admin` ASC),
  CONSTRAINT `fk_post_usuario_creador`
    FOREIGN KEY (`email_creador`)
    REFERENCES `jflorit_prueba_blog`.`usuario` (`email`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_post_usuario_admin`
    FOREIGN KEY (`email_admin`)
    REFERENCES `jflorit_prueba_blog`.`usuario` (`email`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

- *Inserción de los datos.*

```
-- -----------------------------------------------------
-- Data for table `jflorit_prueba_blog`.`usuario`
-- -----------------------------------------------------
START TRANSACTION;
USE `jflorit_prueba_blog`;
INSERT INTO `jflorit_prueba_blog`.`usuario` (`email`, `username`, `password`, `create_time`, `status`)
VALUES ('joan.g.florit@gmail.com', 'joanflo', MD5('password1'), NULL, 'a');
INSERT INTO `jflorit_prueba_blog`.`usuario` (`email`, `username`, `password`, `create_time`, `status`)
VALUES ('cgonzalez@kitmaker.com', 'cgonzalez', MD5('password2'), NULL, 'n');
INSERT INTO `jflorit_prueba_blog`.`usuario` (`email`, `username`, `password`, `create_time`, `status`)
VALUES ('xcunill@kitmaker.com', 'xcunill', MD5('password3'), NULL, 'n');

COMMIT;
```

```
-- --------------------------------------------------------
-- Data for table `jflorit_prueba_blog`.`post`
-- --------------------------------------------------------
START TRANSACTION;
USE `jflorit_prueba_blog`;
INSERT INTO `jflorit_prueba_blog`.`post` (`id_post`, `titulo`, `contenido`, `create_time`, `status`, `email_creador`, `email_admin`)
VALUES (1, 'Lorem ipsum 1', 'Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin auctor urna non arcu blandit mattis. Aenean tristique purus diam, at luctus augue faucibus ac. Etiam in sapien interdum, imperdiet ipsum in, facilisis massa. Suspendisse ut condimentum purus. Donec molestie dapibus lacus, et bibendum nisi lobortis vel. In nulla est, tempus ac felis eget, pharetra ornare magna. Donec non lorem lacus. Nunc blandit, ipsum sed facilisis convallis, nibh mi sollicitudin augue, et condimentum leo est sed velit. Integer vitae auctor odio. Donec posuere, ipsum et tincidunt scelerisque, elit magna convallis neque, ac gravida massa risus eget lacus. Aliquam blandit nulla nec luctus tempor. Duis rutrum pulvinar sapien vel mollis.', NULL, 'p', 'joan.g.florit@gmail.com', 'joan.g.florit@gmail.com');
INSERT INTO `jflorit_prueba_blog`.`post` (`id_post`, `titulo`, `contenido`, `create_time`, `status`, `email_creador`, `email_admin`)
VALUES (2, 'Lorem ipsum 2', 'Donec ut elit sit amet nibh facilisis gravida. Vestibulum pretium augue eu tortor luctus placerat. Curabitur at diam diam. Donec libero nunc, imperdiet vel nulla ut, bibendum dignissim lacus. Maecenas eu eros magna. Donec suscipit libero urna, ut pretium lacus tristique sit amet. Aliquam sem ex, porta non tincidunt et, dignissim sed diam. In faucibus urna vitae aliquam feugiat. Quisque vehicula enim et libero pharetra aliquam. Nullam quis mollis arcu, cursus aliquam massa. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Ut ac elementum velit. Ut eget laoreet orci. Praesent malesuada luctus neque sit amet auctor. Sed consequat, lectus vestibulum fermentum sollicitudin, felis urna vehicula tellus, in elementum nunc augue vel nulla.', NULL, 'p', 'cgonzalez@kitmaker.com', 'joan.g.florit@gmail.com');
INSERT INTO `jflorit_prueba_blog`.`post` (`id_post`, `titulo`, `contenido`, `create_time`, `status`, `email_creador`, `email_admin`)
VALUES (3, 'Lorem ipsum 3', 'Sed nibh dui, varius in placerat vel, tincidunt sit amet ex. Maecenas vehicula mi nunc, nec semper arcu gravida sit amet. Nunc et lorem dolor. Curabitur consectetur lacus a ipsum viverra, sit amet posuere orci dignissim. Curabitur malesuada, tortor dictum viverra tincidunt, erat est volutpat eros, id ultrices lacus nisi sit amet ante. Phasellus arcu neque, consequat eu eros ut, efficitur pharetra augue.', NULL, 'p', 'xcunill@kitmaker.com', 'joan.g.florit@gmail.com');
INSERT INTO `jflorit_prueba_blog`.`post` (`id_post`, `titulo`, `contenido`, `create_time`, `status`, `email_creador`, `email_admin`)
VALUES (4, 'Lorem ipsum 4', 'Quisque eu nisl nec enim porta tempor et in neque. Aenean rhoncus tortor ut lacus posuere tincidunt. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras scelerisque dictum lorem sit amet bibendum. Aliquam et scelerisque dui. Duis mauris ipsum, pulvinar sed semper ac, finibus elementum velit. Curabitur vestibulum porttitor metus in euismod. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Donec et accumsan nibh.', NULL, 'b', 'cgonzalez@kitmaker.com', NULL);
INSERT INTO `jflorit_prueba_blog`.`post` (`id_post`, `titulo`, `contenido`, `create_time`, `status`, `email_creador`, `email_admin`)
VALUES (5, 'Lorem ipsum 5', 'Sed faucibus urna metus, nec congue elit aliquam eu. Sed porta consectetur tellus quis fermentum. In hac habitasse platea dictumst. Vivamus posuere enim eget vestibulum laoreet. Mauris eu magna mattis, scelerisque nibh non, rhoncus mi. Nunc et tellus rutrum, pellentesque nisi id, consectetur purus. Suspendisse potenti.', NULL, 'c', 'xcunill@kitmaker.com', 'joan.g.florit@gmail.com');

COMMIT;
```

2.2 Arquitectura proyecto

Codeigniter es un framework PHP que hace uso del MVC, por lo que en esta prueba se ha respetado dicho patrón:

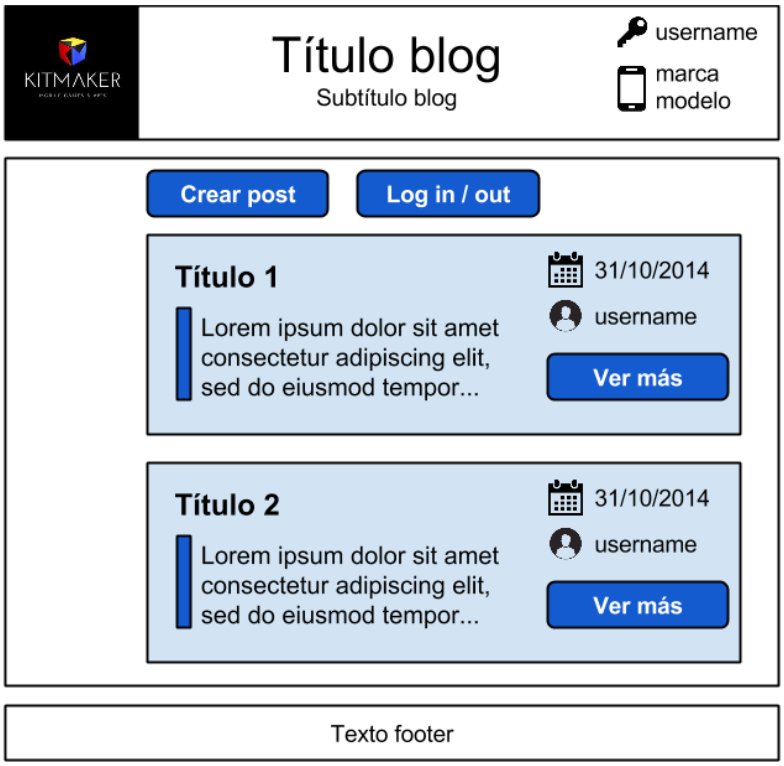
- **Modelos.** Se ha creado uno por cada entidad en la base de datos (posts_model, usuarios_model). Implementan las operaciones con la BDD.
- **Vistas.** Algunas son plantillas (footer, header y navigation_bar) y otras las pantallas que componen el blog (create, index, view).
- **Controlador.** Existe un único controlador (blog) que media entre vistas y modelos.

También se ha creado una librería (Wurflutils) para poder ser reutilizada por múltiples controladores en distintos proyectos.

Los métodos de cada archivo php se encuentran documentados en el código fuente.

2.3 UX (versión desktop)

- Pantalla de inicio con todos los posts públicos (excepto para los admin).



- Pantalla para mostrar un solo post.

- Pantalla que permite crear un nuevo post.

3 Entorno de producción

3.1 Requerimientos

La última versión estable de CodeIgniter (la 2.2) requiere que el servidor:

- disponga de una versión de PHP igual o superior a la 5.1.6.
- que en caso de usar el SGDB MySQL, que la versión de éste sea la 4.1 o superior.

Si además usamos WURFL Cloud, la versión PHP debe ser igual o superior a la 5.3.

En este caso concreto, se ha usado un Apache con PHP 5.5.9 y MySQL 5.5.

3.2 Instalación

1. Configurar el framework.
   - Indicar la URL base en 'application/config/config.php'.
   - Rellenar los datos para poder efectuar la conexión a la BDD en 'application/config/database.php'.
     - La base de datos, de no existir, puede generarse con el script facilitado.
   - En el entorno final de producción, deshabilitar el modo debug de 'index.php'.
2. Configurar WURFL.
   - Registrarse en Scientia Mobile para obtener una key para su API WURFL Cloud, si no se tiene ya una.
   - Habilitar las capabilities que usaremos ('is_mobile', 'brandname', 'modelname').
   - Descargar el cliente para PHP y situarlo en el directorio 'application/third_party'.
   - La librería 'Wurflutils.php' para reutilizar WURFL en múltiples proyectos se encuentra en 'application/libraries'. Referenciar correctamente el cliente e introducir la API key.
3. Subir al servidor el código fuente.
   - Situar los directorios 'application' y 'system' en un directorio que no sea accesible directamente desde el navegador.
   - Situar el 'index.php' en la raíz del directorio público y actualizar la localización de 'application' y 'system'.
   - Se ha añadido un directorio 'assets' con los css, imágenes y javascripts usados por el blog. En el directorio también se encuentra jQuery-UI 1.11.2 (que incluye jQuery 2.1.1), que deberán ser referenciados en las páginas donde se usen. El directorio 'assets' debe ir en la parte pública.

En el siguiente enlace puede verse el blog funcionando en un entorno de producción siguiendo los anteriores pasos: http://alumnes-ltim.uib.es/~jflorit/prueba_blog/index.php/posts

Si surge alguna duda durante el proceso mándese un email a joan.g.florit@gmail.com.

4 Desarrollo del alcance y roadmap

4.1 Alcance

Se han implementado todos los requisitos que figuran en el apartado 1.3, los cuales se desprenden del enunciado de la prueba.

4.2 Roadmap

Estimación por semanas de las funcionalidades a implementar suponiendo un solo programador dedicado:

- 1ª semana (06 – 07 noviembre 2014)
  - Implementar el registro de usuarios, consulta de los datos de un usuario y que los administradores puedan cambiar su rol. Añadir paginación para ver los posts. (2 días)

- 2ª semana (10 – 14 noviembre 2014)
  - Modificar el diseño de la base de datos para permitir nuevos requisitos. Implementar búsqueda de posts por nombre y contenido. Compartir un post en redes sociales. (1 día)
    - Posts clasificados en categorías.
    - Posts con mensajes y likes dejados por los usuarios.
    - Posts con tags.
    - Usuarios que sigan los posts de otros.
    - Mejorar editor para crear posts y poder añadir contenido multimedia.
  - Implementar los anteriores requisitos. (4 días)