Δchan

Proyecto fin de ciclo
Desarrollo de Aplicaciones Multiplataforma

Alfredo Rodríguez Gracía

21 de mayo de 2021

última revisión
24 de mayo de 2021

Índice

1. Descripción del proyecto

Δchan (pronunciado como *dichan*) es un proyecto de tablón de imágenes [4], centrado en el anonimato y la libertad de expresión *on-line*, dónde los usuarios pueden subir imágenes y vídeos cortos para iniciar un debate. Está inspirado en otros tablones existentes como *4chan* y *2channel*, sitios que, a pesar del enorme auge de las redes sociales, siguen siendo el refugio de muchos internautas hoy en día.

Un tablón de imágenes (también conocido por su nombre en inglés: *imageboard*) es un tipo de página web anónima donde la publicación de imágenes y pequeños vídeos cobra una gran importancia. Los primeros tablones de imágenes fueron creados en Japón a finales de los 90, y se basan en el concepto de los foros de texto. En términos generales ambos comparten la misma estructura, incluyendo la separación de los debates (*threads*) de diferentes temáticas en secciones, llamadas tablones o *boards*. Sin embargo, los *threads* en los *imageboards* pueden llegar a ser mucho más esporádicos que en los foros convencionales, donde el tiempo de vida de uno puede ser inferior a varias horas. Los tablones de imágenes más populares en occidente tienden a estar relacionados en su mayoría con la cultura japonesa, como son la temática del *anime* y *manga*. Sin embargo, en Japón son más populares y sus tópicos abarcan una gran variedad de temas.

El proyecto Δchan intenta emular a estos tablones haciendo muy sencillo que cualquiera que lo desee pueda montar su propia instancia en un equipo, incluso con muy pocos recursos. La estructura de la página es muy simple, consta principalmente de dos partes bien diferenciadas: la portada, donde se visualizará la lista de *boards* activos en la página; y los *boards* en sí, cada uno de su temática particular y limitado a nueve páginas de contenido. Cada página de un *board* contendrá cinco *threads* ordenados por fecha de actualización mas reciente, es decir, en el primer puesto de la primera página se colocará el *thread* que ha recibido el último comentario y en el último puesto de la novena página estará el *thread* que ha pasado mas tiempo sin comentarios. En el momento que un usuario decida abrir un nuevo *thread* ese último se borrará y el nuevo aparecerá en el primer puesto. De esta forma se consigue ese dinamismo tán característico de los *imageboards* dónde tienes la certeza de que lo primero que ves al entrar es de lo que se está hablando actualmente, es el tema del momento.

La intención del proyecto mira hacia un futuro colaborativo, donde muchas personas puedan aportar sus opiniones y mejoras al mismo. Este es el motivo por el que se publica bajo la *GNU General Public License version 3* [1], para garantizar que forme parte del movimiento del *software libre* definido por M. Stallman [2]. El código fuente será accesible desde un repositorio *Git*

de libre acceso, donde cualquier persona podrá proponer cambios a través de los procedimientos establecidos. De esta forma también es posible que surjan *forks* [3] donde, utilizando este proyecto como base, cualquiera puede cambiar radicalmente el propósito original y aportar un enfoque completamente diferente.

1.1. Partes del proyecto

- Frontend
  - Página web
- Backend
  - API
  - Base de datos

2. Ámbito de implantación

Deberá describirse el lugar (empresa, organización, sector...) en el que se implantará el proyecto y con qué objetivo, además de indicar a quién va dirigida la aplicación, es decir, identificar quién o quiénes serán los principales usuarios de la misma.

Target: Usuarios a quien les importan mas las ideas, el debate en sí, y no las personas que sostienen esas ideas. El anonimato hace que la gente se pueda expresar sin tapujos y sin esperar consecuencias.

3. Recursos de hardware y software

Se describirán los requisitos mínimos y los requisitos recomendados de hardware, tanto para el desarrollo de la aplicación, como para su instalación y ejecución.

Se describirán las necesidades de software requeridas para el desarrollo de la aplicación.

Puesto que Δchan es un *aplicación web* existen tres escenarios a tratar con respecto a los requisitos de hardware y software: *desarrollo*, *despliegue* e *instalación por parte del usuario*. Inicialmente se pretende que en cualquier de ellos, estos recursos, sean lo más limitados y gratuitos posibles haciendo

que el proyecto pueda ser accesible por cualquiera sin importar la calidad del hardware disponible.

3.1. Requisitos de desarrollo

Muy pocos.

3.2. Requisitos de despliegue

Algunos mas.

Tanto el diseño como el desarrollo de la base de datos se realizarán pensando en un servidor *MariaDB*, dejando así la puerta abierta a una mayor compatibilidad con *MySQL* y otras tecnologías similares. Estará pensada para ser lo mas ligera posible, permitiendo que pueda ser alojada en el mismo equipo que el servidor *HTTP*. Pero siempre permitiendo que pueda instalarse en un equipo diferente, no cerrando la puerta a incrementar las capacidades de la instancia para soportar mayores cantidades de tráfico.

3.3. Requisitos de instalación por parte del usuario

Firefox, HTML5.

4. Temporalización del desarrollo

Deben describirse las distintas actividades necesarias para desarrollar el proyecto, asignarles un tiempo a cada una de ellas y construir los dos diagramas completos.

Tareas a realizar:

1. Empezar
2. Acabar

4.1. Diagrama de Gantt

Pufff...

4.2. Diagrama PERT

Meh...

5. Descripción de los datos base y resultados

Se describirán el tipo de campo (en caso de java serían: String, char, int, double, long...), que se utilizará para recoger los diferentes datos.

Posibles restricciones y/o estructuras utilizadas (clases). Lo mismo para los datos resultantes de los procesos.

Texto introductorio...

5.1. Descripción de la base de datos

El diseño de la base de datos, siempre enfocado hacia una mayor eficiencia y rapidez en la lectura y escritura de los datos, está dividida en tres tablas: *BOARD*, *THREAD* y *POST*. Este diseño intenta compartimentar al máximo las diferentes partes de la página con el mínimo acoplamiento entre la s tablas para evitar hacer *joins* innecesarios, que supondrían una mayor carga para el servidor.

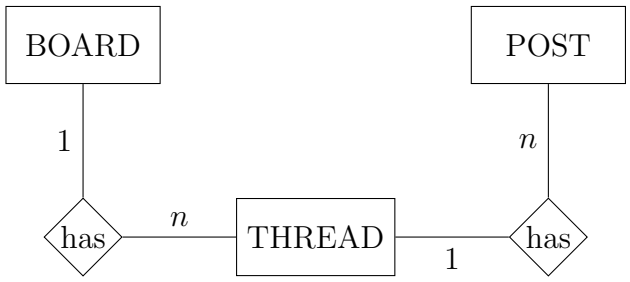

Figura 1: Diagrama Entidad-Relación de la base de datos de Δchan

En el diagrama *ER* de la figura 1 no se han dibujado los atributos de las entidades con el fin de profundizar en ellos a continuación con el mayor detalle posible.

BOARD Cada uno de los tablones que dividen la página en diferentes temas.

- **id** BIGINT PRIMARY KEY
- **name** VARCHAR(256) NOT NULL
- **slug** VARCHAR(3) NOT NULL

THREAD Cada uno de los debates, o *hilos* de discusión, que se inician dentro de un tablón.

- **id** BIGINT PRIMARY KEY
- **subject** VARCHAR(256) DEFAULT '' NOT NULL
- **author** VARCHAR(50) DEFAULT 'Anonymous' NOT NULL
- **comment** VARCHAR(512) DEFAULT 'Anonymous' NOT NULL
- **fileurl** VARCHAR(512) DEFAULT NULL
- **timestamp** DATETIME DEFAULT NOW() NOT NULL
- **sticky** BOOLEAN DEFAULT FALSE NOT NULL
- **closed** BOOLEAN DEFAULT FALSE NOT NULL
- **deleted** BOOLEAN DEFAULT FALSE NOT NULL
- **board** BIGINT FOREIGN KEY REF. POST(id) NOT NULL

POST Cada comentario de un usuario dentro de un debate (*thread*), formado por un texto y una foto o pequeño vídeo opcional.

- **id** BIGINT PRIMARY KEY
- **author** VARCHAR(50) DEFAULT 'Anonymous' NOT NULL
- **comment** VARCHAR(512) DEFAULT 'Anonymous' NOT NULL
- **fileurl** VARCHAR(512) DEFAULT NULL
- **timestamp** DATETIME DEFAULT NOW() NOT NULL
- **deleted** BOOLEAN DEFAULT FALSE NOT NULL
- **thread** BIGINT FOREIGN KEY REF. THREAD(id)

Ejemplo 1: Procedimiento almacenado para la inserción de un nuevo *post*

```
1 DELIMITER $$
2   DROP PROCEDURE IF EXISTS insert_post;$$
3   CREATE PROCEDURE insert_post(
4     IN new_author VARCHAR(50),
```

```
5       IN new_comment VARCHAR(512),
6       IN new_fileurl VARCHAR(512),
7       IN new_thread BIGINT
8     ) BEGIN
9       SET @last_id = 0;
10      SELECT id INTO @last_id FROM POST
11        ORDER BY id DESC LIMIT 1;
12      IF (@last_id IS NULL) THEN
13        SET @last_id = 0;
14      END IF;
15      INSERT
16        INTO POST (id, author, comment, fileurl, thread)
17        VALUES (@last_id + 1, new_author, new_comment,
              new_fileurl, new_thread);
18    END;$$
19  DELIMITER ;
```

5.2. Resultados

Resultados de algunos de los trozos de código que muestro, por ejemplo, algún *JSON* que devuelva la *API*.

¿Capturas de pantalla con ejemplos del funcionamiento de la web, formulario para crear thread, post; portada, lista de boards; el contenido de un thread de ejemplo? ¿Puede que esto vaya en el siguiente apartado? (Preguntar a MJ)

6. Relación entre dispositivos y programa o rutinas

Se identificarán los componentes que comunican el paquete o aplicación software desarrollado con el resto de actores relevantes fuera de la máquina. Es decir, interfaces persona-máquina para entrada y/o salida de datos, interfaces de red u otros medios para comunicación con máquinas remotas, periféricos específicos o componentes concretos de plataformas móviles, etc.

Se identificarán los componentes software (clases, procedimientos) representativos y se vincularán con los anteriormente mencionados a través de texto y/o diagrama(s) que ayuden a comprender el funcionamiento general de la aplicación.

Texto introductorio...

Ejemplo 2: Las funciones de ejemplo

```
/*
 * upperOrNot: Given a character and its predecessor,
 * it returns that same character converted to upper
 * or lower case based on the ASCII value of its
 * predecessor.
 */
const upperOrNot = (previous, actual) => {
  if (previous.charCodeAt(0) % 2 !== 0) {
      return actual.toUpperCase();
  }
  return actual;
}

/*
 * strToNumber: Given a character string returns a
 * string of numbers, based on the value of the
 * characters in the ASCII table.
 */
const strToNumber = (str) => {
  return [...str].map(
    (char) => char.charCodeAt(0) % 10
  ).join('');
}

export { upperOrNot, strToNumber };
```

Referencias

[1] Free Software Foundation Inc. *GNU General Public License version 3*. 2007. URL: https://www.gnu.org/licenses/gpl-3.0.html.

[2] Richard M. Stallman. «3. La definición del software libre». En: *Software libre para una sociedad libre*. Traficantes De Sueños, 2004. URL: https://www.gnu.org/philosophy/fsfs/free_software2.es.pdf.

[3] Wikipedia. *Fork (software development)*. n.d. URL: https://en.wikipedia.org/wiki/Fork_(software_development).

[4] Wikipedia. *Imageboard*. n.d. URL: https://en.wikipedia.org/wiki/Imageboard.

Bloques de código