Sumario

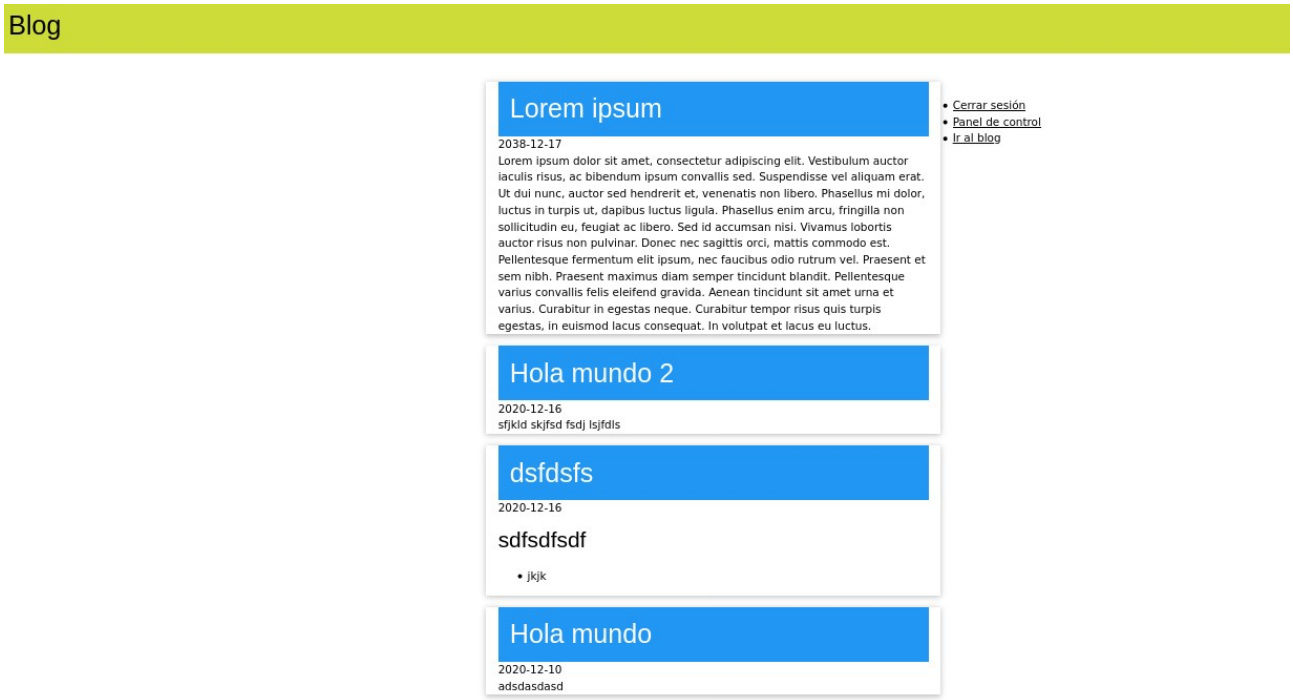
1 Programación de un servicio web con express

El objetivo de la siguiente práctica es realizar un servicio web usando express. Se escribirá una aplicación que servirá para escribir un blog.

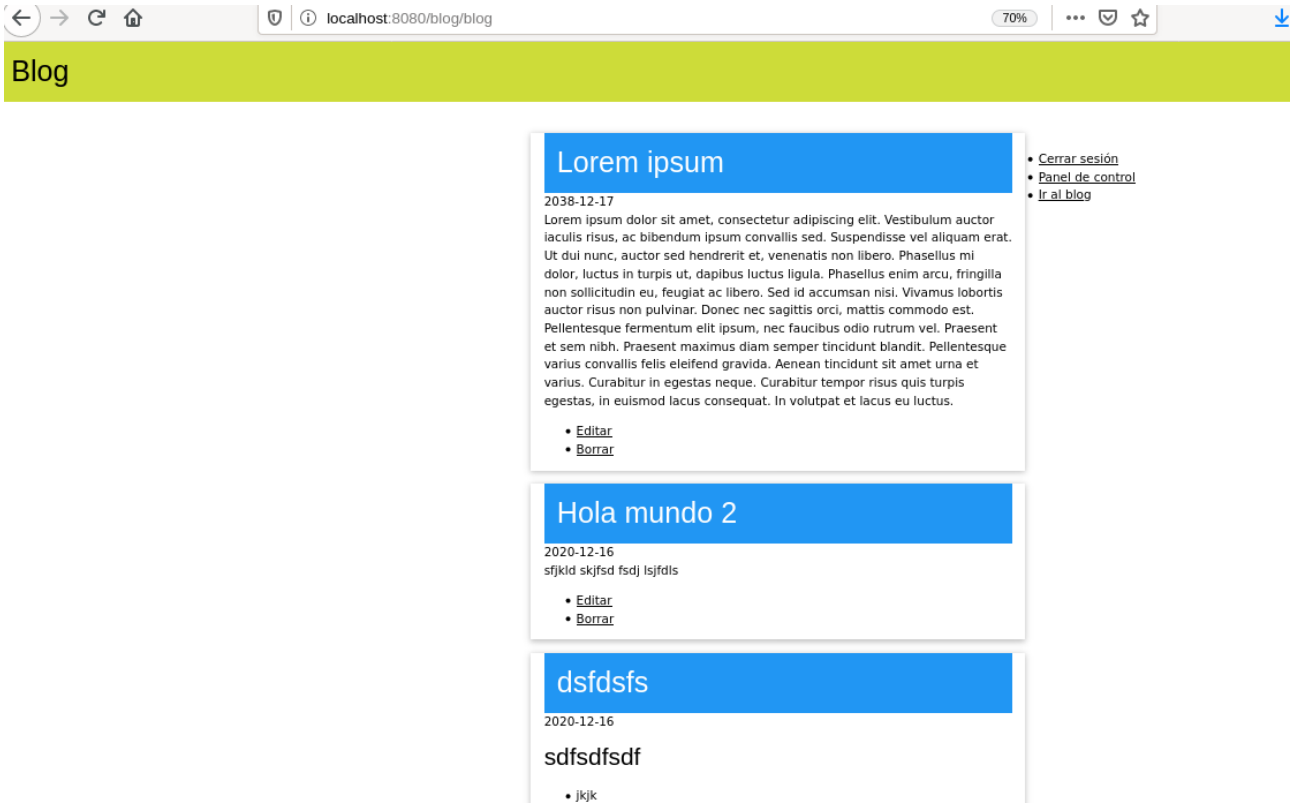
El blog permitirá escribir entradas con un título y un texto que serán mostradas en la página principal.

2 Descripción de la interfaz gráfica

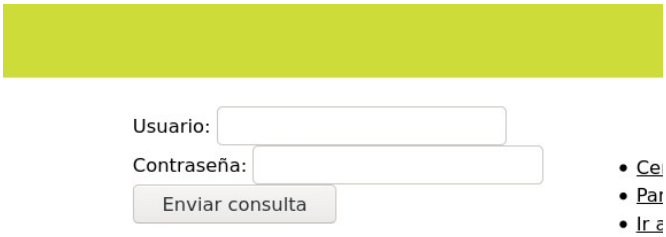
La página que visitará el usuario para ver las entradas será similar a (puede cambiar el diseño por otro más vistoso, este diseño es sólo para hacerse una idea):



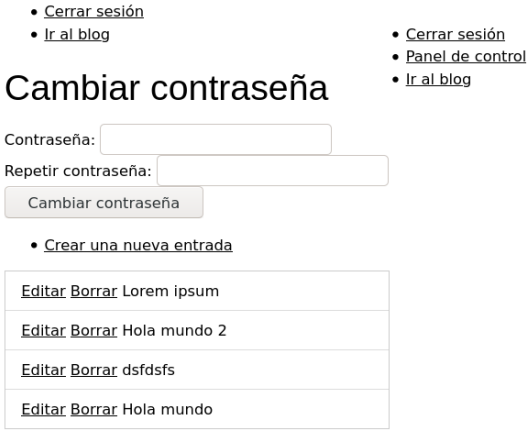
Los usuarios podrán entrar a ver las entradas sin necesidad de autentificarse. A los usuarios autentificados, les aparecerán unas opciones para editar o borrar las entradas existentes:



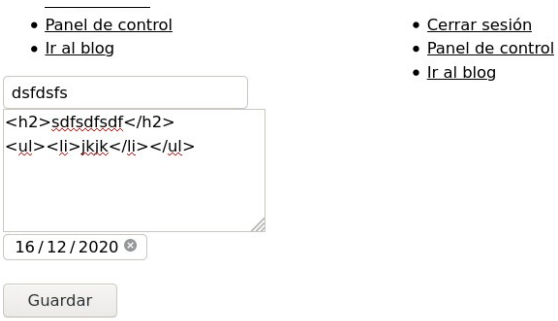
Existirá un panel de control que, para acceder a él, se solicitará que el usuario se autentifique:



Una vez autentificado, el usuario podrá ver un panel de control en el que podrá cambiar su contraseña, crear una nueva entrada o tendrá un listado de las entradas en el que podrá editar o borrar cada entrada:



Si se edita o se crea una nueva entrada, la interfaz será similar a:



3 Descripción de la base de datos

La base de datos es muy simple constando de las siguientes tablas:

CREATE TABLE IF NOT EXISTS **usuarios** (usuario TEXT PRIMARY KEY, password TEXT)

CREATE TABLE IF NOT EXISTS **entradas** (

    id INTEGER PRIMARY KEY AUTOINCREMENT,
    titulo TEXT,
    texto TEXT,
    fecha INTEGER

)

Nota: Como puede verse la fecha es un INTEGER esto se debe a que las fechas se deben almacenar como un número usando la hora UNIX (n.º de segundos que han pasado desde el 1 de enero de 1970).

3.1 Algunas orientaciones respecto a la base de datos

Se usará SQLite para la gestión de los datos. Lea el manual de SQLite en el caso de tener dudas, cada base de datos tiene singularidades.

En el caso de SQLite se puede hacer que asigne la clave primaria de forma automática en el caso de que la clave primaria sea del tipo “INTEGER PRIMARY KEY AUTOINCREMENT”. Para ello se insertan los datos sin insertar la clave primaria. SQLite entenderá que debe crearla él:

INSERT INTO entradas (titulo, texto, fecha) VALUES (?, ?, ?)

Como se puede ver en este INSERT no se indica el id de la clave primaria.

Para obtener el id que SQLite ha asignado a la inserción realizada, se debe realizar una transacción y realizar el siguiente SELECT:

SELECT last_insert_rowid()

Por ejemplo, para consultar el ID del último elemento insertado:

```
const sqlite3 = require('sqlite3')

let db = new sqlite3.Database('./ejer1.db', sqlite3.OPEN_READWRITE |
sqlite3.OPEN_CREATE,
      (err) => {
            if (err) {
                  console.log("Error: " + err)
            } else {
                  crearBase(db);
            }
      }
);

function crearBase(db) {
     db.exec(
     `
      CREATE TABLE IF NOT EXISTS entradas (
      id INTEGER PRIMARY KEY AUTOINCREMENT,
      titulo TEXT,
      texto TEXT,
      fecha INTEGER
        );
     `
     , (err)  => {
        if(err) {
            console.log("Error: " + err);
        } else {
            insertarValores(db)
      }
    });
}

function insertarValores(db) {
      db.all(
      `
            insert into entradas(titulo, texto, fecha) values (?,?,?)
      `,
```

```
            'Hola', 'mundo', 12345,
            (err) => {
                  consulta(db)
            }
      )
}

function consulta(db) {
      db.all(
      `
      select last_insert_rowid()
      `, (err, rows) => {
            if(err) {
                  console.log("Error: " + err)
            } else {
                  rows.forEach( (row) => {
                        let id = row['last_insert_rowid()']
                        console.log(id)
                  })
            }
      })
}
```

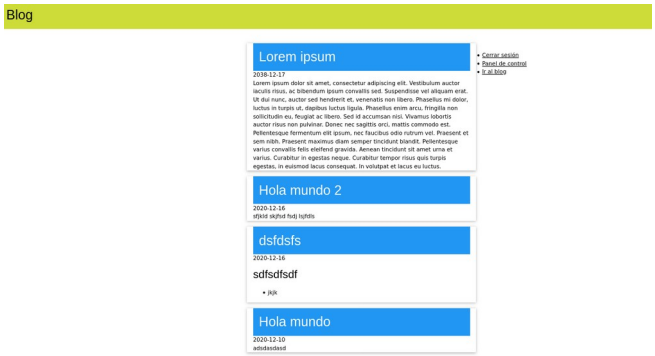
4 Diseño de la aplicación

El código se puede diseñar de forma que los archivos a usar serán similares a:

- views: Contiene plantillas y métodos para generar el HTML a mostrar en las páginas. Es necesario poner un método por página web a desarrollar.
- DB: Representa la conexión a la base de datos. Contiene métodos con los que realizar las consultas, inserciones,… El resto de clases no deben contener código SQL, todas sus consultas se realizarán creando métodos en este archivo.

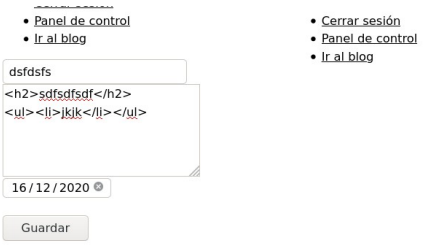
Entre los routes necesarios:

- /Instalador: Creará las bases de datos e insertará un usuario por defecto: el usuario **admin** con contraseña **admin**.
- /InicioSesión: Debe mostrar la página web que se usa para la autentificación y verificar que el usuario ha introducido correctamente sus credenciales.
- /Cerrar: Cierra la sesión abierta por el usuario.
- /Blog: Muestra las entradas del blog. Ejemplo:



- /Panel: Muestra el panel de control en el caso de que el usuario esté autentificado.

- /Editar: Permite editar una entrada:



- … : Los routes que sean necesarios para añadir todas las funciones a la aplicación.

5 Formularios: consideraciones

A la hora de crear los formularios es bueno consultar los tipos de input de los que se dispone en la página:

https://www.w3schools.com/html/html_form_input_types.asp

Por ejemplo, se pueden usar para introducir contraseñas sin que se vean o para introducir fechas:

```
<input type="date" name="fecha">
```

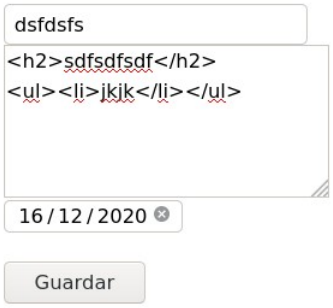
A veces es necesario introducir campos en el formulario que no sean mostrados al usuario. Por ejemplo, en nuestro caso en el panel de control se pueden editar las entradas, cada entrada debe tener un botón, llamado “Editar” que al pulsarlo, lanzará una petición post para mostrar la página de edición. Es decir, cada botón “Editar” serán un formulario con el botóon enviar y el resto de campos ocultos.

Para insertar un campo oculto en un formulario, se usarán los campos hidden:

```
<form method='post' action='/Editar'>
<input type="hidden" name="id_entrada" value="12">
<input type='submit' value='Editar'>
</form>
```

5.1 Inserción de información en los formularios

Durante el desarrollo de la práctica necesitará introducir datos de las entradas en los formularios HTML. Si en los datos hay HTML, se puede producir un mal funcionamiento del código final de la página web. Por ejemplo, suponga que el usuario ha introducido HTML en el editor de la entrada:



Como se puede ver en el TEXTAREA el usuario ha introducido texto que contiene HTML. Supongamos que el usuario ha introducido el siguiente texto:

```
<ul><li>jkjk</li></ul>
</textarea>
Más texto
```

En un primer envío todo funcionará correctamente y se almacenará en la base de datos. Cuando el usuario vuelva a querer editar la página, si no se tiene cuidado al insertar el texto en el TEXTAREA, se producirá la siguiente salida:

Plantilla original:

```
<form method='post' action='editor'>
…
<textarea>
Aquí van los datos
</texarea>
…
</form>
```

HTML generado:

```
<form method='post' action='editor'>
…
<textarea>
<ul><li>jkjk</li></ul>
</textarea>
Más texto
</texarea>
…
</form>
```

Como se puede ver en el HTML generado queda un “</textarea>” desparejado.

En el siguiente ejemplo el usuario introduce unas comillas y un mayor que:

Plantilla original:

```
<input type='hidden' name='id_entrada'
name="Aquí van los datos">
```

HTML generado:

```
<input type='hidden' name='id_entrada'
name="El usuario pone unas comillas y un
mayor que "> ">
```

Como se puede ver la parte roja del HTML generado quedará errónea.

Por lo tanto, antes de introducir los datos habrá que sustituir todos los elementos de HTML que puedan generar problemas. Hay 5 símbolos que deben ser sustituidos:

& → &

> → >

< → <

' → '

" → "

Además deben ser sustituidos por el orden indicado (primero los &, después el >, …).

También hay que tener en cuenta que hay situaciones en las que puede ser conveniente no reemplazar estos símbolos y que se genere el HTML que ha introducido el usuario, por ejemplo, en la página que ya muestra las entradas del blog para su lectura, respetar los datos originales del usuario puede ayudar a enriquecer las entradas al mostrar un texto más vistoso.

6 Fases para la realización de la práctica

La práctica debe realizarse en las siguientes fases. Cada fase puntúa hasta un punto en función de su grado de consecución.

6.1 Fase 1: Configuración e Instalador

En esta práctica se deberá instalar express, las bibliotecas de gestión de las plantillas, cookies y sesiones.

También deberá introducir la biblioteca de SQLite.

Después cree los esqueletos de las views y los archivos DB e Instalador.

Asegúrese de que el Instalador crea las bases de datos e introduce al usuario admin con contraseña admin.

6.2 Fase 2: Autentificación

Ahora se va a crear la route InicioSesión, completando las plantillas y DB con el código que sea necesario para que InicioSesión funcione.

No olvide poner un tiempo máximo a la sesión.

También se deberá implementar la clase Cerrar que cierra la sesión.

Este código es la autentificación básica que se usa en muchos proyectos.

6.3 Fase 3: El panel de control

Modifique InicioSesión para que redireccione a la página del panel. Cree la clase Panel e introduzca el panel de control que se ha indicado en el diseño de la interfaz.

6.4 Fase 4: El Editor

Cuando el usuario pulse la opción de “Crear una nueva entrada” en el panel, se debe abrir el Editor vacío. Cree la clase Editor que sea capaz de abrirse con una entrada vacía o bien buscar una entrada de la cual se le pase como parámetro de una petición GET el id de la página.

Cuando se pulse el botón guardar, la entrada se debe insertar en la base de datos o actualizarse en el caso de que ya existiera.

Verifique que las entradas que se añadan aparezcan en el listado del Panel y se puedan editar o borrar (tendrá que implementar la clase Borrar que borra una entrada).

6.5 Fase 5: El Blog

Deberá implementar la clase Blog que muestra las entradas ordenadas por fecha (primero la más nueva) de una forma vistosa. Debe mostrar las opciones de borrar y editar si el usuario ha iniciado sesión.

6.6 Fase 6: Completar el panel de control

Se completa ahora el panel de control de forma que el usuario pueda cambiar su contraseña.

También, si el usuario es admin, se le mostrará en el panel de control una opción que le conducirá a otra página para gestionar los usuarios, pudiendo dar de alta a nuevos usuarios, cambiarles la contraseña o borrarlos. El resto de usuarios hacen lo mismo que el usuario admin, salvo la posibilidad de acceder a la gestión de usuarios.

6.7 Fase 7: Verificaciones

Verifique que:

1. La sesión tiene un tiempo limitado de duración una vez iniciada.
2. Que las peticiones GET se guarden en caché o no y su fecha de caducidad en el caso de que sí se guarden.
3. Que los datos que introduzca el usuario no rompan los formularios.
4. Verificar que los usuarios no autentificados no puedan acceder al panel, al editor, a la gestión de usuarios,… es decir, las páginas que requieren estar autentificado.
5. Verificar que todas las páginas muestran la codificación correcta.
6. El blog tiene que ser funcional y tener enlaces para llegar a las distintas opciones.
7. Usar campos hidden en los formularios que así lo requieran

6.8 Fase 8: Aspecto y CSS

Haga que el aspecto de las páginas sea visualmente atractivo. Puede usar plantillas como Bootstrap, MaterializeCSS o “w3.css” para mejorar el aspecto de las páginas.

6.9 Fase 9: Seguridad

Es un fallo de seguridad guardar las contraseñas en texto plano. Busque en Internet la forma de generar un código MD5 a partir de las contraseñas. En la base de datos se almacena el MD5 de la contraseña y no la contraseña en sí. La forma de realizar la autentificación es:

1. Se recibe la contraseña que ha escrito el usuario.
2. Se hace el MD5 de la misma.
3. Se compara este MD5 con el MD5 que está almacenado en la base de datos para la contraseña del usuario.
4. Si los MD5 son iguales, se inicia la sesión del usuario.

6.10 Fase 10: Añadir una nueva funcionalidad

Implemente una, **sólo una**, de las siguientes funcionalidades:

- Paginación: Se deben mostrar un máximo de 3 entradas por página. Para ver más entradas habrá que ir a la siguiente página
- Etiquetas: Añadir un campo más en el editor de la entrada, llamado etiqueta. En este campo el usuario sólo puede introducir una palabra. La etiqueta se mostrará al mostrar la entrada y, cuando se pulse en ella, se mostrarán todas las entradas que tengan la misma etiqueta
- Comentarios: Usuarios sin autentificar deben poder escribir comentarios en cada una de las entradas.