Flujo de Trabajo del CRUD en Laravel

En este ejercicio, hemos desarrollado un CRUD (Crear, Leer, Actualizar, Eliminar) básico para gestionar comentarios en una aplicación Laravel.

La persistencia de los comentarios se realiza en la sesión utilizando un array de strings. A continuación, se detalla el flujo de trabajo del

CRUD implementado y se presentan los códigos de los archivos 'web.php' y 'CommentController.php' con comentarios explicativos.

Flujo de Trabajo:

1. Listado de Comentarios (GET /comments):

Se llama al método `index()` del controlador, que recupera los comentarios almacenados en la sesión y los devuelve como respuesta JSON.

2. Formulario de Creación (GET /comments/create):

Se llama al método `create()`, que retorna una vista con un formulario HTML para añadir nuevos comentarios.

3. Almacenar Comentario (POST /comments):

Se llama al método `store()`, que toma el comentario de la solicitud y lo añade al array de comentarios en la sesión, devolviendo el array actualizado.

4. Mostrar Comentario (GET /comments/{commentid}):

Se llama al método `show()`, que devuelve el comentario especificado por su índice en el array.

5. Formulario de Edición (GET /comments/{commentid}/edit):

Se llama al método `edit()`, que retorna una vista con un formulario para editar el comentario especificado.

6. Actualizar Comentario (PATCH /comments/{commentid}):

Se llama al método `update()`, que modifica el comentario en la posición especificada del array en la sesión.

7. Eliminar Comentario (DELETE /comments/{commentid}):

Se llama al método `destroy()`, que elimina el comentario especificado del array en la sesión.

Código de web.php con Comentarios:

```
<?php

use Illuminate\Support\Facades\Route;
use App\Http\Controllers\CommentController;

// Ruta para listar los comentarios
Route::get('/comments', [CommentController::class, 'index']);

// Ruta para mostrar el formulario de creación de un nuevo comentario
Route::get('/comments/create', [CommentController::class, 'create']);

// Ruta para almacenar un nuevo comentario
Route::post('/comments', [CommentController::class, 'store']);

// Ruta para mostrar un comentario específico
Route::get('/comments/{commentid}', [CommentController::class, 'show']);

// Ruta para mostrar el formulario de edición de un comentario existente
Route::get('/comments/{commentid}/edit', [CommentController::class, 'edit']);
```

```
// Ruta para actualizar un comentario existente
Route::patch('/comments/{commentid}', [CommentController::class, 'update']);

// Ruta para eliminar un comentario
Route::delete('/comments/{commentid}', [CommentController::class, 'destroy']);
```

Código de CommentController.php con Comentarios:

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class CommentController extends Controller
{
    // Método para listar todos los comentarios
    public function index()
    {
        // Recupera los comentarios de la sesión
        $comments = session('comments', []);
        return response()->json($comments);
    }


    // Método para mostrar el formulario de creación de un comentario
    public function create()
    {
        // Retorna la vista para crear un comentario
```

```
    return view('comments.create');
}

// Método para almacenar un nuevo comentario
public function store(Request $request)
{
    // Obtiene el comentario del request y lo añade a la sesión
    $comment = $request->input('comment');
    $comments = session('comments', []);
    $comments[] = $comment;
    session(['comments' => $comments]);
    return response()->json($comments);
}

// Método para mostrar un comentario específico
public function show($commentid)
{
    $comments = session('comments', []);
    // Verifica si el comentario existe en la posición especificada
    if (!isset($comments[$commentid])) {
        return response()->json(['error' => 'Comentario no encontrado'], 404);
    }
    return response()->json($comments[$commentid]);
}

// Método para mostrar el formulario de edición de un comentario
public function edit($commentid)
{
    $comments = session('comments', []);
```

```
        // Verifica si el comentario existe en la posición especificada
        if (!isset($comments[$commentid])) {
            return response()->json(['error' => 'Comentario no encontrado'], 404);
        }
        // Retorna la vista para editar un comentario
        return view('comments.edit', ['comment' => $comments[$commentid],
'commentid' => $commentid]);
    }

    // Método para actualizar un comentario
    public function update(Request $request, $commentid)
    {
        $comments = session('comments', []);
        // Verifica si el comentario existe en la posición especificada
        if (!isset($comments[$commentid])) {
            return response()->json(['error' => 'Comentario no encontrado'], 404);
        }
        // Actualiza el comentario en la posición especificada
        $comments[$commentid] = $request->input('comment');
        session(['comments' => $comments]);
        return response()->json($comments);
    }


    // Método para eliminar un comentario
    public function destroy($commentid)
    {
        $comments = session('comments', []);
        // Verifica si el comentario existe en la posición especificada
```

```
        if (!isset($comments[$commentid])) {
            return response()->json(['error' => 'Comentario no encontrado'], 404);
        }
        // Elimina el comentario de la posición especificada
        unset($comments[$commentid]);
        session(['comments' => $comments]);
        return response()->json($comments);
    }
}
```

Interacción Cliente-Servidor en el CRUD de Comentarios:

1. Cliente: El cliente inicia la interacción enviando una solicitud a través de un navegador o una herramienta como cURL.

   Esta solicitud puede ser para listar comentarios, crear un nuevo comentario, editar o eliminar uno existente.

2. Servidor (Laravel): El servidor Laravel recibe la solicitud y la redirige a la ruta correspondiente definida en el archivo `web.php`.

   Cada ruta está asociada a un método específico en el `CommentController`.

3. Rutas: Las rutas son la primera capa de la aplicación que conecta la solicitud del cliente con la lógica del servidor.

   Cada ruta en `web.php` define el método HTTP (GET, POST, PATCH, DELETE) y la URL que debe coincidir para ejecutar un método específico del controlador.

4. Controlador (CommentController): El controlador actúa como un intermediario entre las rutas y las vistas.

Contiene la lógica para manejar las solicitudes del cliente. Por ejemplo, si un cliente solicita crear un nuevo comentario,

la ruta `/comments` con método POST llama al método `store()` en el `CommentController`, que gestiona la lógica para almacenar el comentario en la sesión.

5. Vistas: Las vistas son responsables de la presentación y la interfaz de usuario. En este CRUD, las vistas muestran formularios HTML para la creación y edición de comentarios (`create.php` y `edit.php`).

Las vistas son retornadas por los métodos `create()` y `edit()` del controlador y presentan la interfaz para que el usuario interactúe con los datos.

6. Respuesta al Cliente: Una vez que el controlador ha procesado la solicitud, devuelve una respuesta al cliente.

Esta respuesta puede ser un JSON con los datos actualizados, un formulario HTML, o un mensaje de error si la operación no se pudo completar.