Práctica 1 UF3. La biblioteca de Barcelona.

Enunciado

Dar de alta un libro telemáticamente se está convirtiendo en toda una pesadilla para los trabajadores de una biblioteca localizada en la ciudad de Barcelona ubicada en el barrio de Sant Antoni.

Por ello se les ha pedido a los alumnos de DAW2 la realización de una aplicación en Laravel que permita gestionar internamente el archivo de libros almacenados, para facilitar las gestiones a los trabajadores de la biblioteca que usen la web.

Como segunda entrega, los alumnos trabajarán simulando una metodología **Scrum** y presentarán al cliente **diversas operaciones con QueryBuilder de Laravel** en un segundo *sprint* que durará **una semana**.

La entrega deberá hacerse en un repositorio en **GitLab.**

El profesor actuará de representante técnico de la biblioteca para valorar si lo solicitado por la biblioteca está presente en la aplicación de Laravel, le pedirá usar rollbacks y ejecutar migraciones-seeders respectivamente.

Funcionamiento

QueryBuilder

1. Crea una aplicación Laravel con dos modelos: "Book" y "Category". Los modelos deben estar relacionados mediante una relación muchos a muchos.
2. Crea un controlador llamado "BookController" con los siguientes métodos:

    - **index**: Devuelve una vista que muestra todos los libros con su información básica (título, autor, imagen, etc.).

- **show**: Devuelve una vista que muestra la información detallada de un libro específico.
- **create**: Devuelve una vista con un formulario para crear un nuevo libro.
- **store**: Crea un nuevo libro en la base de datos a partir de los datos enviados por el formulario.
- **edit**: Devuelve una vista con un formulario para editar la información de un libro específico.
- **update**: Actualiza la información de un libro específico en la base de datos a partir de los datos enviados por el formulario.
- **destroy**: Elimina un libro específico de la base de datos.

Crea los siguientes endpoints en el archivo de rutas "web.php":

- **`GET /books`**: Llama al método "index" del controlador "BookController".
- **`GET /books/{id}`**: Llama al método "show" del controlador "BookController".
- **`GET /books/create`**: Llama al método "create" del controlador "BookController".
- **`POST /books`**: Llama al método "store" del controlador "BookController".
- **`GET /books/{id}/edit`:** Llama al método "edit" del controlador "BookController".
- **`PUT /books/{id}`:** Llama al método "update" del controlador "BookController".
- **`DELETE /books/{id}`:** Llama al método "destroy" del controlador "BookController".

4. Agrega un filtro por categoría en la vista de "index" y "show". Para ello, debes agregar una lista desplegable con las categorías disponibles y, al seleccionar una categoría, se debe mostrar solo los libros que pertenecen a esa categoría.
5. Implementa la lógica necesaria en el controlador y las vistas para agregar, editar y eliminar categorías para los libros.
6. Agrega una paginación en la vista de "index" para mostrar solo un número limitado de libros por página.

    - Podemos usar el método **paginate(nºelementos por página)**

- Habrá que interpolar en la vista el método **links()** que proporcionará un enlace a cada nº de página correspondiente.

7. Implementa la funcionalidad de búsqueda de libros en la vista de "index". Para ello, debes agregar un formulario de búsqueda y mostrar solo los libros que coinciden con los términos de búsqueda ingresados por el usuario.

Entrega

Se valorará positivamente:

- **Identificación, declaración y uso de clases para el problema.**
- **Limpieza y orden en la estructura.**

Al hacer las subidas cuida los elementos que hay que añadir al **.gitignore** para no incluir información sensible (ej: configuración bbdd) de tu proyecto o información en exceso (dependencias /vendor) en el repositorio remoto.

La entrega se hará dejando el **enlace del repositorio de Gitlab** con todo el código PHP, el fichero de configuración y el fichero log.