

[<--- Volver](#)

Filtering

Restricciones de consulta elocuente avanzada / Advanced Eloquent Query Constraints

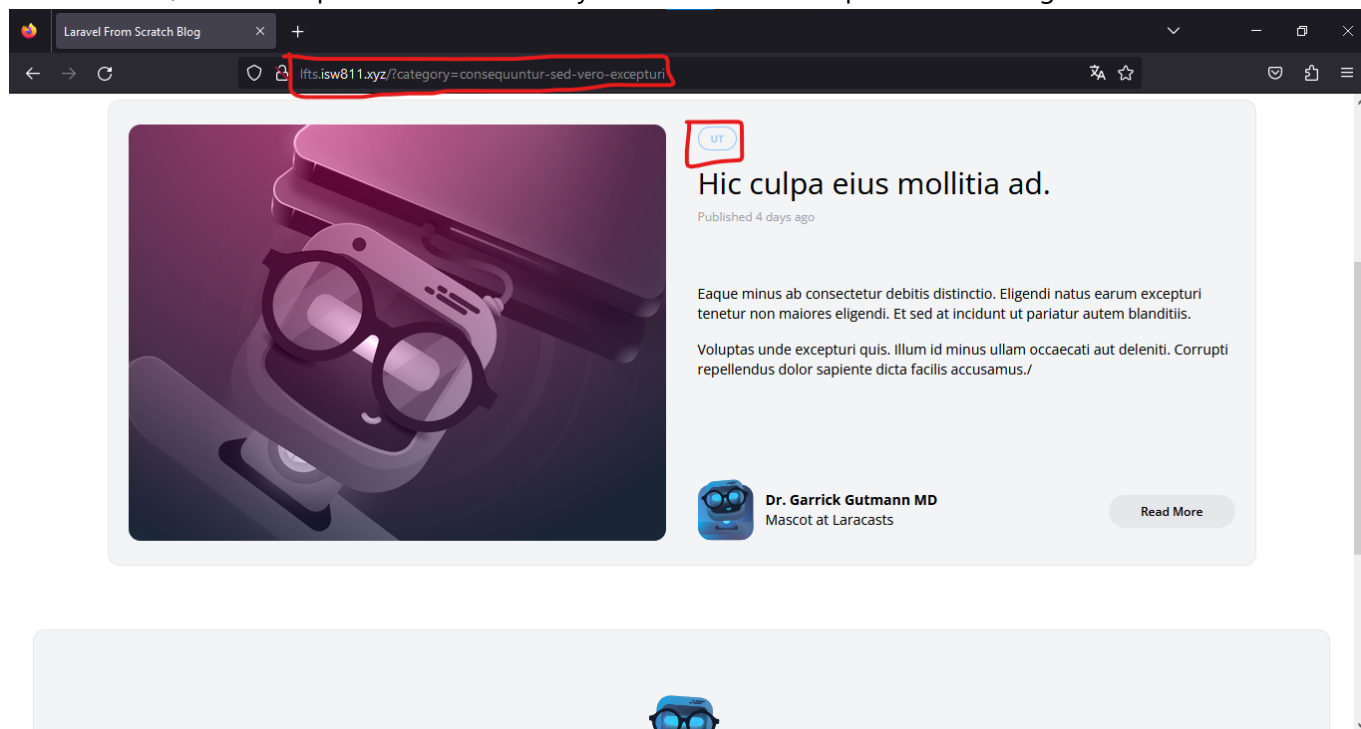
Modificamos en PostController la función index para que también se habilite la búsqueda por categorías

```
public function index() {  
  
    return view('posts', [  
        'posts' => Post::latest()->filter(request(['search', 'category']))->get(),  
        'categories' => Category::all()  
    ]);  
}
```

Y añadimos un nuevo filtro en la función scopeFilter() en la clase Post

```
public function scopeFilter($query, array $filters) {  
  
    $query->when($filters['search'] ?? false, fn($query, $search) =>  
        $query->where('title', 'like', '%' . $search . '%')  
            ->orWhere('body', 'like', '%' . $search . '%'));  
  
    $query->when($filters['category'] ?? false, fn($query, $category) =>  
        $query  
            ->whereExists(fn($query) =>  
                $query->from('categories')  
                    ->whereColumn('categories.id', 'posts.category_id')  
                    ->where('categories.slug', $category))  
            );  
}
```

Como vemos, buscamos por medio de la URL y este nos muestra el post con la categoria correcta.



En este caso la categoria y el slug no son los mismos pero si revisamo en workbench veremos que ese slug equivale a esa categoria

name	slug
ut	consequuntur-sed-vero-excepturi

Otra manera un poco mas limpia de utilizar el mismo query es la siguiente

```
public function scopeFilter($query, array $filters) {

    $query->when($filters['search'] ?? false, fn($query, $search) =>
        $query->where('title', 'like', '%' . $search . '%')
        ->orWhere('body', 'like', '%' . $search . '%'));

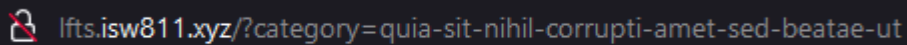
    $query->when($filters['category'] ?? false, fn($query, $category) =>
        $query
            ->whereHas('category', fn ($query) =>
                $query->where('slug', $category)
            )
    );

}
```

Ahora vamos a post-header y vamos a actualizar el metodo por el que buscamos las categorias

```
<x-dropdown-item
    href="/?category={{ $category->slug }}"
    :active="request()->is('categories/' . $category->slug)"
    >{{ ucwords($category->name) }}
</x-dropdown-item>
```

Ahora volvemos a la pagina y buscamos por medio de categorias y veremos que la URL será el nuevo metodo y no el antiguo



En PostController a la funcion index agregamos lo siguiente

```
public function index() {  
    return view('posts', [  
        'posts' => Post::latest()->filter(request(['search', 'category']))->get(),  
        'categories' => Category::all(),  
        'currentCategory' => Category::firstWhere('slug', request('category'))  
    ]);  
}
```

Por ultimo en web.php eliminamos la ruta de las categorías por lo que el archivo quedaría así

```
Route::get('/', [PostController::class, 'index'])->name('home');  
  
Route::get('posts/{post}', [PostController::class, 'show']);  
  
Route::get('authors/{author:username}', function (User $author) {  
    return view('posts', [  
        'posts' => $author->posts,  
        'categories' => Category::all()  
    ]);  
});
```

Extraer un componente de hoja desplegable de categoría / Extract a Category Dropdown Blade Component

Lo primero que haremos será crear un nuevo componente llamado `category-dropdown` lo podemos crear de la siguiente manera desde la terminal de nuestra VM webserver en la ruta de nuestro proyecto

```
php artisan make:component CategoryDropdown
```

Pegamos toda la logica del dropdown dentro de este nuevo componente

Dentro de `app/View/Components` se creó un nuevo componente al que agregaremos el siguiente código

```
public function render()  
{
```

```

        return view('components.category-dropdown', [
            'categories' => Category::all()
        ]);
    }

```

Modificamos le archivo web.php ya que no necesitamos pasar las categorias por la ruta

```

Route::get('/', [PostController::class, 'index'])->name('home');

Route::get('posts/{post}', [PostController::class, 'show']);

Route::get('authors/{author:username}', function (User $author) {

    return view('posts', [
        'posts' => $author->posts
    ]);
});

```

Y en PostController modificamos también

```

public function index() {

    return view('posts', [
        'posts' => Post::latest()->filter(request(['search', 'category']))->get()
    ]);
}

```

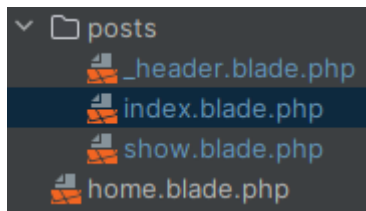
Movemos el trozo de código de que eliminamos de PostController y lo movemos al componente de CategoryDropdown ya que éste será quien se encargue de las categorías

```

public function render()
{
    return view('components.category-dropdown', [
        'categories' => Category::all(),
        'currentCategory' => Category::firstWhere('slug', request('category'))
    ]);
}

```

Dentro de la carpeta *resources/views* vamos a crear una carpeta llamada *posts* para guardar las vistas de los post y renombramos las vistas



Esto para que se llamen como sus respectivas funciones dentro del PostControllerS

Filtro de autores / Author Filtering

Lo primero que haremos será ir a post-card y post-featured-card buscar donde se carga el nombre del autor para convertirlo en un link

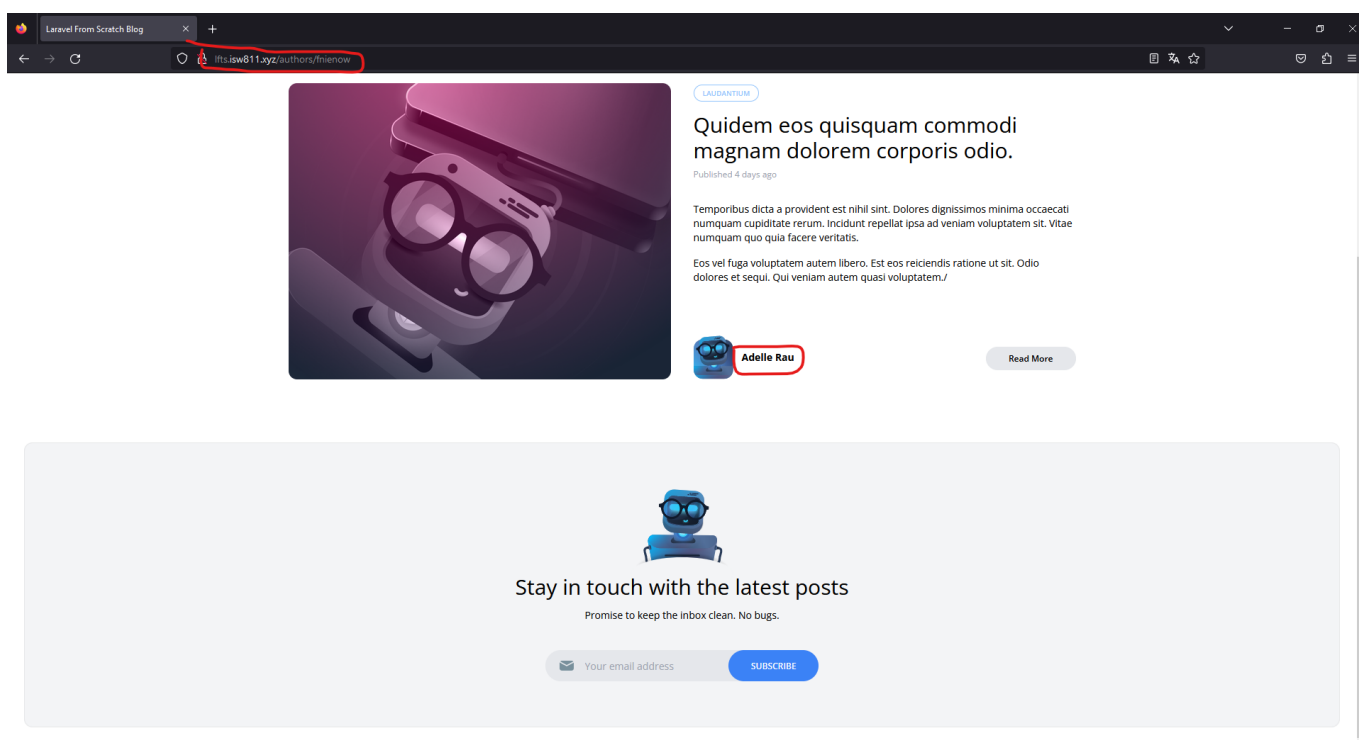
```
<h5 class="font-bold">
<a href="/authors/{{ $post->author->username }}">{{ $post->author->name }}</a>
</h5>
```

Luego en el archivo de rutas modificamos lo cambiado anteriormente en PostController

```
Route::get('authors/{author:username}', function (User $author) {

    return view('posts.index', [
        'posts' => $author->posts
    ]);
});
```

Vemos en la web como la darle click al autor nos lleva a una pagina donde están los posts de ese mismo autor



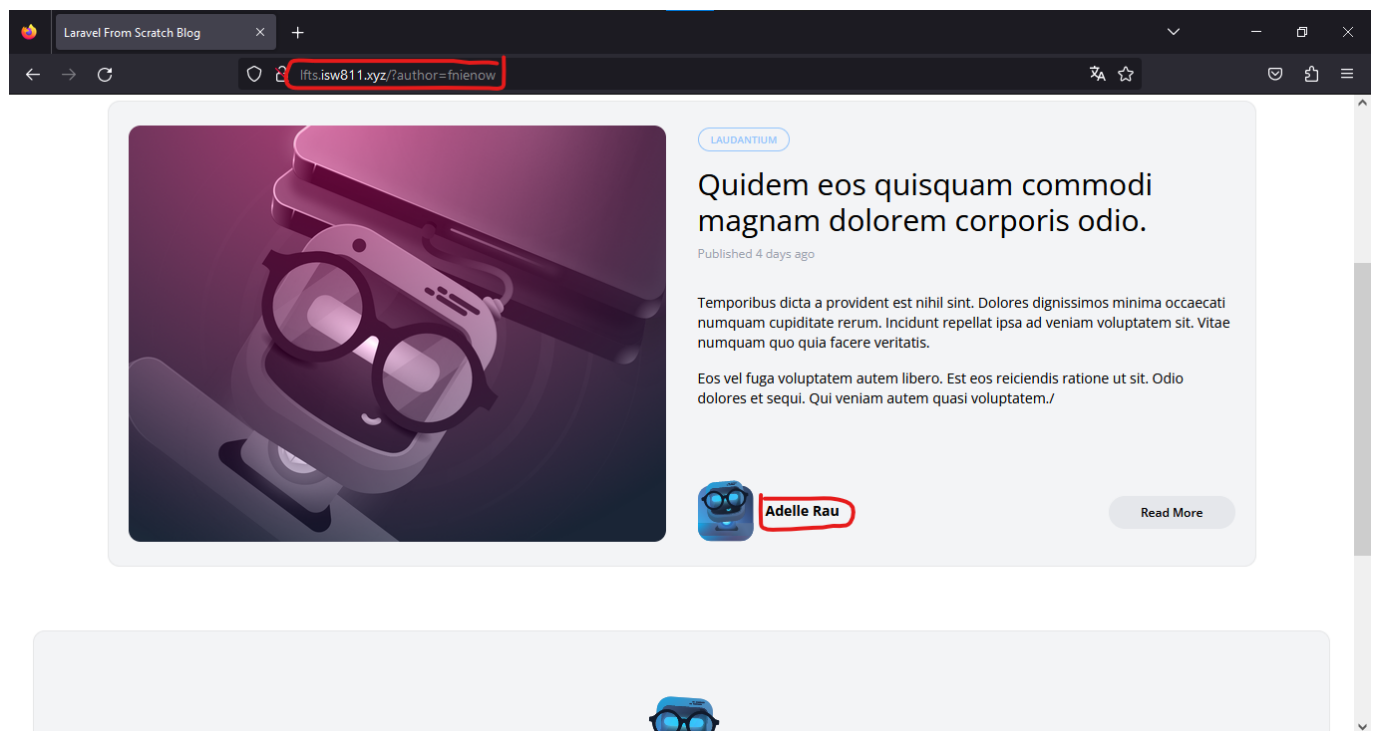
Ahora nos movemos a la clase post para añadir el autor en la funcion filter para posteriormente poder buscar de la misma manera en la que lo hacemos con las categorias

```
$query->when($filters['author'] ?? false, fn($query, $author) =>
    $query
        ->whereHas('author', fn ($query) =>
            $query->where('username', $author)
        )
    );
```

Luego en PostController también agregamos el autor, al igual que lo hicimos con las palabras y las categorias

```
public function index() {
    return view('posts.index', [
        'posts' => Post::latest()->filter(request(['search', 'category',
        'author']))->get()
    ]);
}
```

Luego vamos a la web e intentamos buscar por medio del query de la URL



Iremos a la vista show para que al ingresar al post el nombre del autor también sea un link que nos redirija a los posts de ese autor

Modificamos el url para que busque de la misma manera en la que lo hacemos con las palabras y las categorias

```
{{ $post->author->name }}
```

Y por ultimo eliminamos el endpoint de authors que tenemos en nuestro archivo de rutas ya que no lo necesitaremos

```
Route::get('/', [PostController::class, 'index'])->name('home');

Route::get('posts/{post}', [PostController::class, 'show']);
```

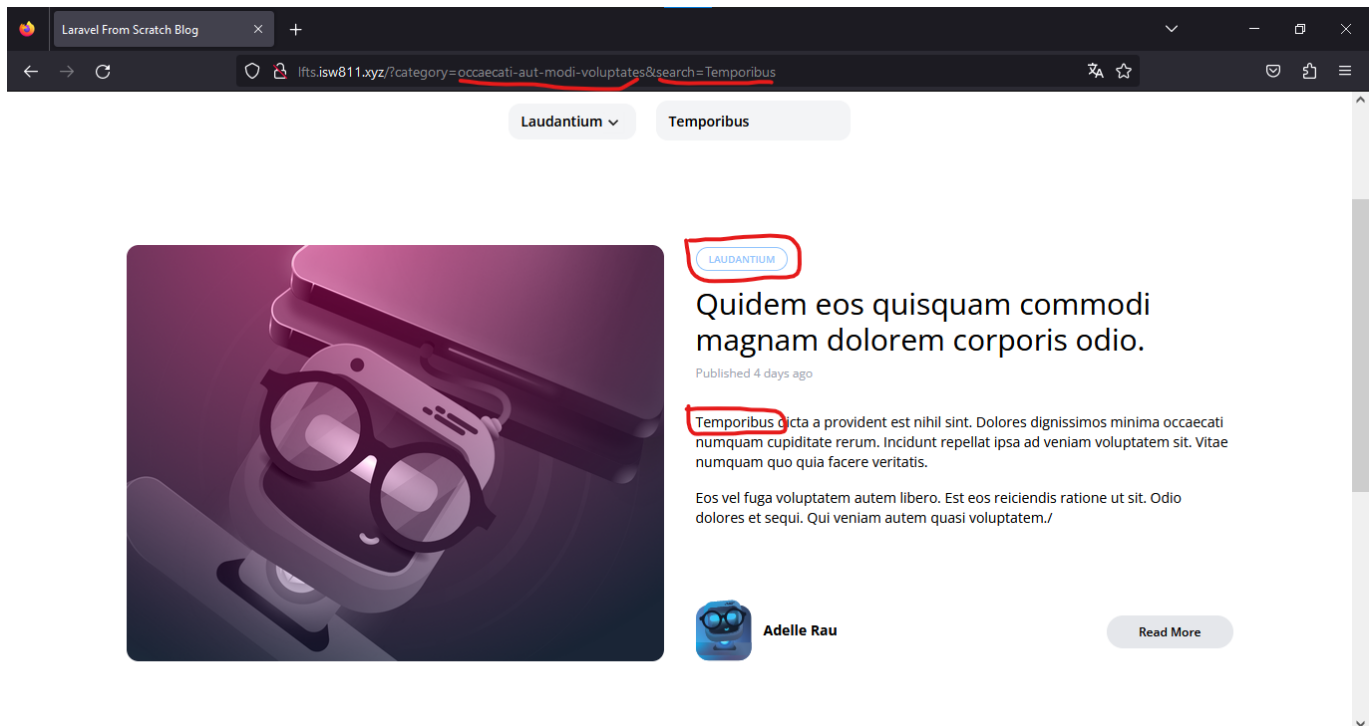
Fusionar categorías y consultas de búsqueda / Merge Category and Search Queries

Cuando buscamos por medio de categorías y luego aplicamos el filtro de palabras este ultimo nos borra la categoría que escogimos, por lo que el filtro buscaría las palabras en todas las categorías, para solucionar eso haremos lo siguiente

Iremos a _header y modificamos

```
<!-- Search -->
<div class="relative flex lg:inline-flex items-center bg-gray-100 rounded-
xl px-3 py-2">
  <form method="GET" action="/">
    @if(request('category'))
      <input type="hidden" name="category" value="{{
request('category' ) }}">
    @endif
    <input
      type="text"
      name="search" placeholder="Find something"
      class="bg-transparent placeholder-black font-semibold text-sm"
      value="{{ request('search' ) }}"
    >
  </form>
```

Como podemos ver en la imagen ambos filtros se estan aplicando en el query string



Ahora haremos que funciona al revés, si hay una palabra filtrada y queremos filtrar también por categoría lo podemos hacer, para esto vamos a la vista `category-dropdown` nuestro

```
<x-dropdown-item
  href="/?category={{ $category->slug }}&{{ http_build_query(request()-
>except('category')) }}"
  :active="request()-is('categories/' . $category->slug)"
  >{{ ucwords($category->name) }}
</x-dropdown-item>
```

Corregir un error de consulta elocuente y confuso / Fix a Confusing Eloquent Query Bug

Hay un pequeño bug en el query que solicionamos de esta manera, vamos a la clase `Post` y modificamos

```
public function scopeFilter($query, array $filters) {
    $query->when($filters['search'] ?? false, fn($query, $search) =>
        $query->where(fn($query) =>
            $query->where('title', 'like', '%' . $search . '%')
                ->orWhere('body', 'like', '%' . $search . '%')
        ));

    $query->when($filters['category'] ?? false, fn($query, $category) =>
        $query
            ->whereHas('category', fn ($query) =>
                $query->where('slug', $category)
            )
    );

    $query->when($filters['author'] ?? false, fn($query, $author) =>
        $query
```



```
    ->whereHas('author', fn ($query) =>
      $query->where('username', $author)
    )
  );
}
```