https://doi.org/10.22201/dgtic.ctud.2025.3.1.100

Vol. 3, Núm. 1. enero-marzo 2025, págs. 36 - 50

Reordenamiento dinámico de elementos usando *drag & drop* en Laravel-11 y Livewire-3

José Antonio Ochoa Rodríguez

Dirección General de Cómputo y de Tecnologías de Información y Comunicación

Universidad Nacional Autónoma de México

jantonio.ochoarz@gmail.com

ORCID: 0009-0006-1559-2346

Información del reporte:

Para citar este reporte técnico:

Ochoa Rodríguez, J. A. (2025). Reordenamiento dinámico de elementos usando drag & drop en Laravel-11 y Livewire-3. *Cuadernos Técnicos Universitarios de la DGTIC, 3 (1)* páginas (36 - 50).

https://doi.org/10.22201/dgtic.ctud.2025.3.1.100

Resumen

Cada categoría y subcategoría de software publicada en el "Portal de Software UNAM" es mostrada al usuario conforme al parámetro "orden_tienda". Este valor se define previamente en el Módulo de Gestión de Categorías y Subcategorías de Software de la herramienta web en desarrollo denominada "Censo TIC". Durante el desarrollo de dicho módulo, se implementó un control de interfaz de usuario *(widget)* que, con sólo arrastrar y soltar elementos de una lista con el puntero del ratón *(widget drag & drop)*, permite actualizar de manera automática el valor de "orden_tienda" dentro de la base de datos; Esto resultó en una interfaz interactiva, fácil de usar y eficiente, que brinda la posibilidad de una menor cantidad de clics, un menor desplazamiento entre distintas pantallas para obtener datos y visualizar toda la información relacionada entre sí de cada categoría y sus subcategorías. El empleo de este tipo de controles permite una mejor experiencia de usuario *(UX)*, enfocada en todo momento en la facilidad de uso, la eficiencia, la utilidad, la satisfacción y la emoción que el usuario experimenta durante la interacción con la interfaz web.

Palabras clave:

Ordenamiento dinámico, lista, *drag & drop*, Laravel, Livewire.

https://doi.org/10.22201/dgtic.ctud.2025.3.1.100 **Vol. 3, Núm. 1.** enero-marzo 2025, págs. 37 - 50

1. INTRODUCCIÓN

La Dirección General de Cómputo y de Tecnologías de Información y Comunicación (DGTIC) de la UNAM puso a disposición de la comunidad universitaria el "Portal de Software de la UNAM", sitio web mediante el cual, entre otras actividades, se difunde, para su descargo e instalación, el software libre y el software adquirido con convenio para apoyar a las funciones sustantivas.

Actualmente, los responsables de TIC de la UNAM requieren del apoyo del personal de la DGTIC para gestionar los elementos publicados en dicho portal, lo que conlleva a tiempos de espera largos, dependencia técnica y velocidad de respuesta poco oportuna. Por tal motivo, la DGTIC inició el desarrollo del componente de software denominado "Censo TIC", plataforma web modular desarrollada con el *framework* de PHP Laravel 11, Livewire 3 y TailwindCSS, que permitirá a los responsables de TIC realizar dichas tareas de forma automatizada y en tiempo real.

Al tratarse de una herramienta modular en desarrollo, fue preciso iniciar con la codificación de los componentes necesarios para los módulos de gestión de "Categorías de software" y "Subcategorías de software". Si bien el prototipo recibido señalaba la creación de módulos independientes, se decidió crear un único componente de software denominado "Módulo de Categorías de Software", que brinda al usuario final una única interfaz web integral que permite el registro, modificación, listado, consulta, activación, desactivación y reordenamiento de aparición "orden_tienda" en el "Portal de Software UNAM" de las categorías y subcategorías de software, de una manera simple, interactiva, coherente y de rápido aprendizaje.

El reto por resolver fue la creación de un control de interfaz de usuario *(widget)* que, con sólo arrastrar y soltar elementos de una lista con el puntero del ratón *(widget drag & drop)*, permitiera actualizar de manera automática el valor de "orden_tienda" dentro de la base de datos, ya sea para una categoría o para una subcategoría contra sus similares pertenecientes a la misma categoría. Esto evita al usuario tener que navegar entre distintas pantallas, el uso de controles de formulario poco intuitivos y hacer varios clics hasta obtener el "orden_tienda" deseado.

El objetivo del trabajo presentado a continuación es el describir el proceso de implementación de un componente de software que permite reordenar los elementos de un listado y actualizar automáticamente el valor dentro de la base de datos con sólo arrastrar y soltar elementos con el puntero del ratón (*widget drag & drop*).

2. DESARROLLO TÉCNICO

Para el desarrollo del módulo, se empleó el repositorio "Censo TIC" creado por la DGTIC, que incluye un contenedor en Docker con los servicios:

- Apache: software que procesa y entrega archivos de sitios web a los usuarios desde un navegador.
- PHP: lenguaje de programación empleado para el desarrollo web y que se ejecuta del lado del servidor.
- Composer: manejador de paquetes para PHP que proporciona un estándar para administrar, descargar e instalar dependencias y librerías.

https://doi.org/10.22201/dgtic.ctud.2025.3.1.100 **Vol. 3, Núm. 1.** enero-marzo 2025, págs. 38 - 50

- Laravel 11: marco o esquema de trabajo de PHP para el desarrollo de software y que simplifica el trabajo repetitivo.
- Livewire 3 para Laravel: herramienta que permite crear aplicaciones interactivas con componentes Javascript para Laravel.
- TailwindCSS: marco o esquema de trabajo de CSS para aplicar estilos a interfaces de usuario HTML.
- PostgreSQL: sistema manejador de bases de datos relacional.

Barajas González (2023) señala que "el uso de contenedores para desarrollo de software facilita al programador la instalación de las herramientas necesarias para aplicaciones cuya arquitectura requiere diversos servicios adicionales. También facilita trabajar sobre sistemas legados porque encapsula los archivos binarios y librerías que estos necesitan para funcionar, en un entorno independiente al sistema operativo que lo contiene" y que "Los contenedores solucionan la incompatibilidad entre ambientes de programación en proyectos con más de un programador ya que pueden crear contenedores con las mismas características, independientemente del sistema operativo o configuraciones propias de cada uno de sus equipos de cómputo".

De acuerdo con Adriano Escudero y Guapi Cuji (2024), "teniendo en cuenta que en el desarrollo web de buscar más eficiencia, confiabilidad, mantenibilidad y escalabilidad, se necesita de la ayuda de un *framework* que sea una parte esencial del conjunto de herramientas de desarrollo de aplicaciones web que utilizan el lenguaje PHP. Varios *framework* de PHP han demostrado ser grandes complementos del desarrollo de software con código a prueba de fallas y extensible, lo que da como resultado aplicaciones web más sólidas y seguras."

Por otra parte, Sánchez Montes de Oca (2024) indica que "Laravel es un *framework* de desarrollo web de código abierto escrito en PHP que se utiliza para crear aplicaciones web de alta calidad y escalables, siendo uno de los *frameworks* de PHP más populares".

Barroso Benítez, Trujillo Casanola y Millet Lombida (2021) señalan que:

> En el desarrollo de software se reconoce la usabilidad como atributo de calidad para el éxito de un producto, no obstante, son pocos los procesos y profesionales que la aplican (Castilla, Hernández y González, 2017). La usabilidad constituye un factor esencial en el desarrollo de los sistemas informáticos (Ferré, 2018), si estos no son percibidos como una herramienta que ayuda al usuario a realizar sus tareas, se dificulta en gran manera la aceptación del mismo. En los últimos años se ha identificado el concepto de "Experiencia de Usuario" (UX por sus siglas en inglés). La UX está tomando mayor importancia a nivel mundial y cada vez más organizaciones se preocupan por las emociones, percepciones y respuestas que tendrán los usuarios cuando interactúen con sus productos y servicios.

En conclusión, la combinación de tecnologías seleccionadas permite desarrollar un sistema técnicamente robusto y sólido, que facilita el trabajo de los desarrolladores y asegura una experiencia de usuario positiva, respondiendo a las demandas actuales de eficiencia, seguridad y usabilidad en los sistemas informáticos.

https://doi.org/10.22201/dgtic.ctud.2025.3.1.100 **Vol. 3, Núm. 1.** enero-marzo 2025, págs. 39 - 50

2.1 METODOLOGÍA

2.1.1 ANÁLISIS

El modelo de entidad-relación de la base de datos indicó las siguientes entidades:

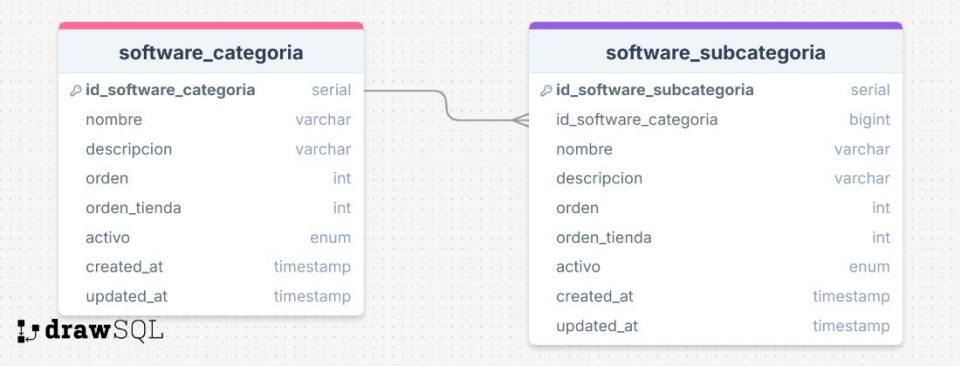
- "software_categoria" para almacenar las categorías de software, con los atributos: id_software_categoria, nombre, descripción, orden, orden_tienda, activo created_at y updated_at.
- "software_subcategoria" para almacenar las subcategorías de software, con los atributos: id_software_subcategoria, id_software_categoria, nombre, descripción, orden, orden_tienda, activo created_at y updated_at.

Podemos observar que el modelo define una relación de tipo "uno a muchos (1:M)"; es decir, que una software_categoria CONTIENE una o muchas software_subcategoria y, a su vez, una software_subcategoria PERTENECE a una software_categoria.

Como se ha definido anteriormente, el reto por resolver fue la creación de un control de interfaz de usuario *(widget)* que, con sólo arrastrar y soltar elementos de una lista con el puntero del ratón *(widget drag & drop)*, permitiera actualizar de manera automática el valor de "orden_tienda" dentro de la base de datos, ya sea para una categoría o para una subcategoría contra sus similares pertenecientes a la misma categoría.

Figura 1

Modelo Entidad-Relación para categorías y subcategorías de software



La documentación del proyecto definió un prototipo que separa en componentes distintos la gestión de software_categoria y software_subcategoria; es decir, "Módulo de Categorías" y "Módulo de Subcategorías"; cada uno de ellos permitiendo el registro, modificación, listado, consulta, paginación, activación, desactivación y reordenamiento de aparición "orden_tienda".

https://doi.org/10.22201/dgtic.ctud.2025.3.1.100 **Vol. 3, Núm. 1.** enero-marzo 2025, págs. 40 - 50

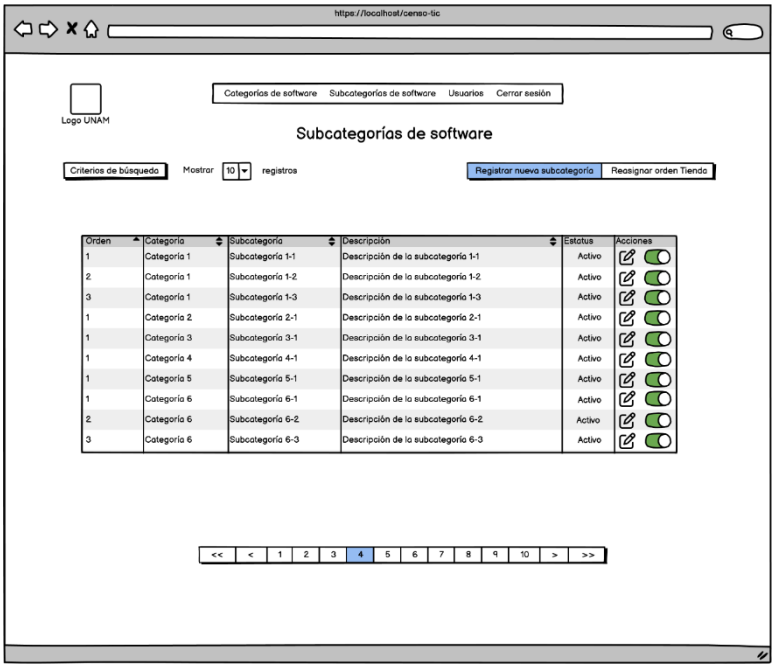
Figura 2

Prototipo inicial para el componente de listado de categorías de software

Y subcategorías de software:

Figura 3

Prototipo inicial para el componente de listado de subcategorías de software



https://doi.org/10.22201/dgtic.ctud.2025.3.1.100 Vol. 3, Núm. 1. enero-marzo 2025, págs. 41 - 50

2.1.2 DISEÑO

Se propusieron las siguientes interfaces de usuario para el listado y reordenamiento de categorías de software y sus subcategorías:

Figura 4

Prototipo final para el componente de listado de categorías de software

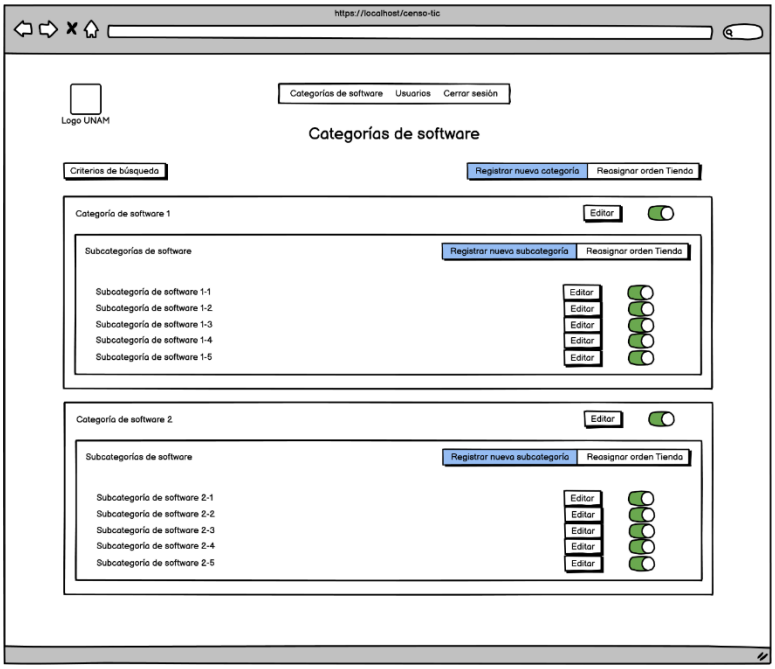
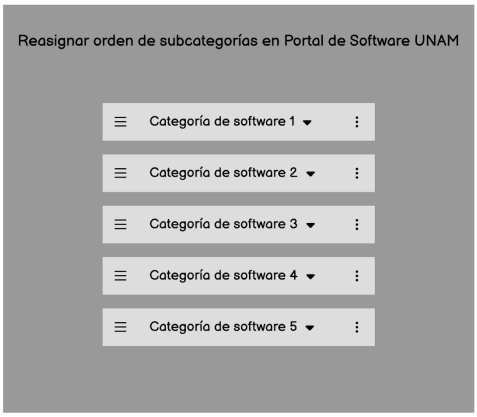


Figura 5

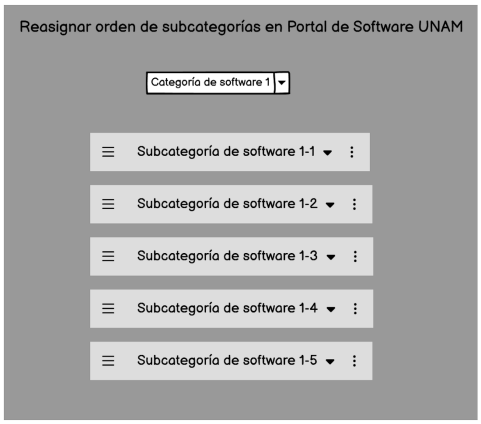
Prototipo final para actualizar el orden de categorías de software de manera dinámica



https://doi.org/10.22201/dgtic.ctud.2025.3.1.100 **Vol. 3, Núm. 1.** enero-marzo 2025, págs. 42 - 50

Figura 6

Prototipo final para actualizar el orden de subcategorías de software de manera dinámica



2.1.3 IMPLEMENTACIÓN

Para implementar la funcionalidad de reordenamiento dinámico de categorías y subcategorías de software mediante un *widget* de arrastrar y soltar (*drag & drop*), que además actualiza automáticamente los valores en la base de datos, se pueden observar los pasos de la implementación en los anexos A y B correspondientes.

2.1.4 PRUEBAS

Se comprobó que las vistas se representan adecuadamente en el navegador, que las acciones de los botones se ejecutan y muestran las ventanas modales esperadas, que al arrastrar un elemento del listado de categorías o subcategorías se mueve conforme se manipula el ratón y que, al soltar el elemento, se dispara la acción sobre la base de datos. Se verificaron los nuevos valores de *"tienda_orden"* en la tabla correspondiente de la base de datos.

2.1.5 MANTENIMIENTO

Al tener bien organizados todos los elementos de ambos componentes, se facilita el mantenimiento y la expansión del Módulo de Categorías y Subcategorías de Software a lo largo del tiempo; sin embargo, será necesario adaptar a los cambios que exijan las nuevas versiones de Laravel y Livewire.

3. RESULTADOS

Al finalizar el módulo de gestión de "Categorías y Subcategorías de software", se entregó una única interfaz web que integra las acciones de registro, modificación, listado, consulta, activación, desactivación y reordenamiento de aparición en el Portal de Software UNAM. Asimismo, se creó un componente que permite el reordenamiento de aparición de las categorías y subcategorías de software en dicho portal, mediante una interfaz sencilla, simple, interactiva, coherente y de rápido aprendizaje. Esto se

https://doi.org/10.22201/dgtic.ctud.2025.3.1.100 **Vol. 3, Núm. 1.** enero-marzo 2025, págs. 43 - 50

logra con sólo arrastrar y soltar elementos con el puntero del ratón (*widget drag & drop*) para actualizar automáticamente el valor de *"orden_tienda"* dentro de la base de datos.

4. CONCLUSIONES

El desarrollo de componentes de software con diseño responsivo, atractivo, interactivo y, sobre todo, funcional permite mejorar la experiencia de usuario y un rápido aprendizaje. Al crear una interfaz que integra las acciones necesarias sobre las categorías y subcategorías de software, se evita el exceso de clics y se logra un menor desplazamiento entre distintas pantallas para obtener información y mantener el enfoque respecto a toda la información relacionada entre sí.

Asimismo, al sustituir los controles numéricos para definir el reordenamiento de aparición de las categorías y subcategorías de software en el Portal de Software UNAM, que implicaría capturar un valor numérico más un clic en el botón de "guardar", por una interfaz que permita arrastrar y soltar elementos de una lista con el puntero del ratón (*widget drag&drop*) para actualizar automáticamente el valor de *"orden_tienda"* dentro de la base de datos, permitimos que dicha experiencia de usuario sea más atractiva, fluida e interactiva.

AGRADECIMIENTOS

A la Doctora Ana Yuri Ramírez Molina, Titular de la Dirección; al Maestro Hugo Alonso Reyes Herrera, Titular de la Subdirección de Sistemas Integrados y a la Licenciada Karla Fonseca Márquez, Titular del Departamento de Integración de Soluciones de Software. Todos ellos de la Dirección de Colaboración y Vinculación dependiente de la Dirección General de Cómputo y de Tecnologías de Información y Comunicación (DGTIC). Gracias por brindarme la oportunidad de cumplir uno de mis más grandes sueños: servir profesionalmente a nuestra máxima casa de estudios; por su enorme apoyo, su guía y orientación; por ser el modelo a seguir, de quienes admiro todo su compromiso y esfuerzo; por su liderazgo, que me lleva a esforzarme por alcanzar mi verdadero potencial.

REFERENCIAS

Adriano Escudero, W. & Guapi Cuji, D.(2023) Influencia del código abierto y su simplicidad en el desarrollo de sistemas web académicos. RCTU [online].vol.10, n.1 [citado 2024-11-15], pp.10-18. Recuperado de: http://scielo.senescyt.gob.ec/scielo.php?script=sci_arttext&pid=S1390-76972023000100010&lng=es&nrm=iso ISSN 1390-7697. https://doi.org/10.26423/rctu.v10i1.730.

Barajas González, L. D. (2023). Uso de contenedores para la construcción de productos de software. *Cuadernos Técnicos Universitarios de la DGTIC*,1(1), pp.72-81. https://doi.org/10.22201/dgtic.ctud.2023.1.1.15

Barroso Benítez, Y., Trujillo Casanola, Y. & Millet Lombida, Y. (2021). Marco de trabajo de evaluación de experiencia de usuario en el desarrollo de software. *Revista cubana de Ciencias Informáticas* [online]. vol.15, n.3 [citado 2024-11-15], pp. 92-117. Recuperado de: http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S2227-18992021000300092&lng=es&nrm=iso Epub 01-Sep-2021. ISSN 2227-1899.

Castilla R, L., Hernández G, D. & González P, Y. (2017). De la arquitectura de información a la experiencia de usuario: Su interrelación en el desarrollo de software de la Universidad de las Ciencias Informáticas.

https://doi.org/10.22201/dgtic.ctud.2025.3.1.100 **Vol. 3, Núm. 1.** enero-marzo 2025, págs. 44 - 50

E-Ciencias de la Información. Vol. 7, 1, pp.155-176.

Laravel (2011 - 2024). Recuperado de https://laravel.com/docs/11.x

Laravel - Artisan Console. (2011 – 2024) Recuperado de https://laravel.com/docs/11.x/artisan

Manual de PHP. (2001-2024). Recuperado de https://www.php.net/manual/es/

Sánchez Montes de Oca, P. Z. (2024). Automatización de tareas para desarrolladores de software en Laravel, por medio de comandos personalizados. *Cuadernos Técnicos Universitarios de la DGTIC, 2* (3), pp.40-49. https://doi.org/10.22201/dgtic.ctud.2024.2.3.61

ANEXO A

Categorías de software

Para implementar la funcionalidad de reordenamiento dinámico de categorías de software mediante un *widget* de arrastrar y soltar (*drag & drop*), que además actualiza automáticamente los valores en la base de datos, se llevaron a cabo los siguientes pasos:

1. Se generó el componente Livewire y la vista Blade con el siguiente comando: `php artisan make:livewire ReasignarOrdenTiendaSoftwareCategoríasComponent`
2. Dentro del componente, se generó el método actualizarOrdenTiendaSoftwareCategorias(), responsable de recibir el arreglo de categorías de software con el nuevo orden:

Figura 7

Código fuente del método actualizarOrdenTiendaSoftwareCategorias()

```
// app/Livewire/SoftwareCategorias/ReasignarOrdenTiendaSoftwareCategoríasComponent.php

public function actualizarOrdenTiendaSoftwareCategorias($softwareCategorias)
{

ReasignarOrdenTiendaSoftwareCategoriasAction::reasignarOrdenTiendaSoftwareCategorias($softwareCate
gorias, Auth::id());
    $mensaje = 'software_categorias.orden_tienda.exito';
    $this->success($mensaje);
}
```

3. Luego, se creó manualmente el archivo responsable de realizar la operación dentro de la base de datos: ReasignarOrdenTiendaSoftwareCategoriasAction.php y, dentro de la transacción a la base de datos, se definió el código que recorre el arreglo de categorías de software con el nuevo orden y ajusta el valor dentro de la base de datos:

https://doi.org/10.22201/dgtic.ctud.2025.3.1.100 **Vol. 3, Núm. 1.** enero-marzo 2025, págs. 45 - 50

Figura 8

Código fuente de la acción ReasignarOrdenTiendaSoftwareCategoriasAction()

```
// Modulos/SoftwareCategorias/Actions

class ReasignarOrdenTiendaSoftwareCategoriasAction
{
    public static function reasignarOrdenTiendaSoftwareCategorias($softwareCategorias, int
$idUsuario)
    {
        try {
              _

              foreach ($softwareCategorias as $softwareCategoria) {
                   SoftwareCategoria::updateOrCreate(
                       [
                           'id_software_categoria' => $softwareCategoria['value'],
                       ],
                       [
                           'orden_tienda' => $softwareCategoria['order'],
                       ]
                   );
               }
            });
        } catch (Exception $e) {
            Log::error($e::class . ' > ' . $e->getFile() . '('.$e->getLine().'): ' .
$e->getMessage());
            throw $e;
        }
    }
}
```

4. Se generó el modelo SoftwareCategoria empleando el comando:

   ```
   php artisan make:model SoftwareCategoria
   ```

   Por default, Laravel 11 generó el archivo *SoftwareCategoria.php* en la ruta *app/Models* y se reubicó manualmente en *modulos/SoftwareCategorias/Models*.

5. Después, se modificó el código fuente de la vista en Blade:

https://doi.org/10.22201/dgtic.ctud.2025.3.1.100 **Vol. 3, Núm. 1.** enero-marzo 2025, págs. 46 - 50

Figura 9

Código fuente de la vista reasignar-orden-tienda-software-categorias.blade.php

```
<!--
resources/views/livewire/software-categorias/reasignar-orden-tienda-software-categorias.blade.php
-->

<div>
    <x-dialog-modal wire:model="modalAbierto">
        <x-slot name="title">
            Reasignar orden de aparición en Portal de Software UNAM
        </x-slot>

        <x-slot name="content">
            <div class="overflow-x-auto">
                <div class="w-96 mx-auto ">
                    <ul wire:sortable="actualizarOrdenTiendaSoftwareCategorias">
                        @foreach ($this->softwareCategorias as $softwareCategoria)
                            <li wire:sortable.item="{{ $softwareCategoria->id_software_categoria
}}" wire:key="categoria-{{ $softwareCategoria->id_software_categoria }}" class="flex
justify-between border rounded mb-3 overflow-hidden">

                                <span wire:sortable.handle class=" grid place-content-center
bg-stone-50 hover:bg-sky-50 px-3 cursor-move rounded-lg ">
                                    <i class="fa-solid fa-grip-vertical "></i>
                                </span>
                                <span class=" w-full  flex justify-between items-center pl-2">
                                    <strong>{{  $softwareCategoria->nombre  }}</strong>
                                    <img src="{{
Storage::url('public/'.$softwareCategoria->imagen) }}" width="40" class=" w-14 h-14">
                                </span>
                            </li>
                        @endforeach
                    </ul>
                </div>
            </div>
        </x-slot>

        <x-slot name="footer">
            <x-primary-button type="button" class="mr-3"
wire:click="cerrarModalReasignarOrdenTienda" target="guardar">
                Actualizar orden
            </x-primary-button>
            <x-secondary-button type="button" class="" wire:click="$toggle('modalAbierto')"
wire:loading.attr="disabled">Cerrar</x-secondary-button>
        </x-slot>
```

6. Por último, se modificó el código fuente del botón “Reasignar orden en Portal de Software UNAM” de las categorías, definido en la vista Blade *listar-software-categorias.blade.php*:

https://doi.org/10.22201/dgtic.ctud.2025.3.1.100 **Vol. 3, Núm. 1.** enero-marzo 2025, págs. 47 - 50

Figura 10

Código fuente del botón "Reasignar orden en Portal de Software UNAM" en la vista listar-software-categorias.blade.php

```
<!-- resources/views/livewire/software-categorias/listar-software-categorias.blade.php -->

@if (count($this->softwareCategorias) > 0)
    @can('reasignar-orden-tienda-software-categorias')
        <div class="flex justify-end">
            <x-action-button class="bg-indigo-600"
wire:click="$dispatch('abrir-modal-reasignar-orden-tienda-software-categorias');"
data-tippy="Reasignar Orden en Portal de software UNAM">
                <i class="fa-solid fa-list-squares"></i>
            </x-action-button>
        </div>
    @endcan
@endif
```

El evento "clic" del botón ejecuta *"abrir-modal-reasignar-orden-tienda-software-categorias"* para mostrar la ventana modal con el listado de Categorías de Software. Al arrastrar y soltar algún elemento del listado, se llama al evento *actualizarOrdenTiendaSoftwareCategorias,* que realiza la actualización en la base de datos de manera dinámica.

ANEXO B

Subcategorías de software

Para implementar la funcionalidad de reordenamiento dinámico de subcategorías de software mediante un *widget* de arrastrar y soltar (*drag & drop*), que además actualiza automáticamente los valores en la base de datos, se llevaron a cabo los siguientes pasos:

1. Se generó el componente Livewire y la vista Blade con el siguiente comando:

```
php artisan make:livewire ReasignarOrdenTiendaSoftwareSubcategoríasComponent
```

2. Dentro del componente, se generó el método actualizarOrdenTiendaSoftwareSubcategorias(), responsable de recibir el arreglo de categorías de software con el nuevo orden:

Figura 11

Código fuente del método actualizarOrdenTiendaSoftwareSubcategorias()

```
public function actualizarOrdenTiendaSoftwareSubcategorias($softwareSubcategorias)
{

ReasignarOrdenTiendaSoftwareSubcategoriasAction::reasignarOrdenTiendaSoftwareSubcategorias($softwa
reSubcategorias, Auth::id());
    $mensaje = 'software_subcategorias.orden-tienda.exito';
    $this->success($mensaje);
}
```

3. Luego, se creó manualmente el archivo responsable de realizar la operación dentro de la base de datos: ReasignarOrdenTiendaSoftwareSubcategoriasAction.php y, dentro de la transacción a la base

https://doi.org/10.22201/dgtic.ctud.2025.3.1.100 Vol. 3, Núm. 1. enero-marzo 2025, págs. 48 - 50

de datos, se definió el código que recorre el arreglo de categorías de software con el nuevo orden y ajusta el valor dentro de la base de datos:

Figura 12

Código fuente de la acción ReasignarOrdenTiendaSoftwareCategoriasAction()

```
// Modulos/SoftwareSubcategorias/Actions

class ReasignarOrdenTiendaSoftwareSubcategoriasAction
{
    public static function reasignarOrdenTiendaSoftwareSubcategorias($softwareSubcategorias, int
$idUsuario)
    {
        try {

            --

             foreach ($softwareSubcategorias as $softwareSubcategoria) {
                 SoftwareSubcategoria::updateOrCreate(
                     [
                         'id_software_subcategoria' => $softwareSubcategoria['value'],
                     ],
                     [
                         'orden_tienda' => $softwareSubcategoria['order'],
                     ]
                 );

                 //Guardado en bitácora
                    Bitacora::registrar($idAccion, $idUsuario, $softwareSubcategoria['value'],
RegistroTipoEnum::SoftwareSubcategoria);
             }
         });
        } catch (Exception $e) {
              Log::error($e::class . ' > ' . $e->getFile() . '('.$e->getLine().'): ' .
$e->getMessage());
            throw $e;
        }
    }
```

4. Se generó el modelo SoftwareSubcategoria empleando el comando:

```
php artisan make:model SoftwareSubcategoria
```

Por default, Laravel 11 generó el archivo *SoftwareSubcategoria.php* en la ruta *app/Models* y se reubicó manualmente en *modulos/SoftwareSubcategorias/Models*.

5. Después, se modificó el código fuente de la vista en Blade

Figura 13

Código fuente de la vista reasignar-orden-tienda-software-subcategorias.blade.php

https://doi.org/10.22201/dgtic.ctud.2025.3.1.100 Vol. 3, Núm. 1. enero-marzo 2025, págs. 49 - 50

```
<!-
resources/views/livewire/software-subcategorias/reasignar-orden-tienda-software-subcategorias.bla
e.php -->


<div>
    <x-dialog-modal wire:model="modalAbierto">
        <x-slot name="title">
            Reasignar orden de aparición en Portal de Software UNAM
        </x-slot>

        <x-slot name="content">
            <div class="overflow-x-auto">
                <div class="mb-3 grid grid-cols-1 gap-2 w-96 mx-auto">
                    <div class="flex flex-col">
                        <x-input-label for="formRegistrarSoftwareSubcategoriaCategoria">Categorí
de software <span class="obligatorio">*</span></x-input-label>
                                                <select   class="w-full   h-auto"   name="rol"
id="formRegistrarSoftwareSubcategoriaCategoria"
wire:change="actualizarListadoSubcategorias($event.target.value)" >
                            @if(!$this->modalDesdeCategoria)
                                <option value="">-- Seleccione una categoría --</option>
                            @endif
                            @foreach ($this->softwareCategorias as $softwareCategoria)
                                <option value="{{ $softwareCategoria->id_software_categoria }}">
                                    {{ ucfirst($softwareCategoria->nombre) }}
                                </option>
                            @endforeach
                        </select>
                        @error('formRegistrarSoftwareSubcategoria.id_software_categoria') <spa
class="error">{{ $message }}</span> @enderror
                    </div>
                </div>
                <div class="mb-3 grid grid-cols-1 gap-2 w-9/12 mx-auto">
                    <ul wire:sortable="actualizarOrdenTiendaSoftwareSubcategorias">
                        @foreach ($this->softwareSubcategorias as $softwareSubcategoria)
                                                                <li    wire:sortable.item="{
$softwareSubcategoria->id_software_subcategoria            }}"            wire:key="categoria-{
$softwareSubcategoria->id_software_subcategoria  }}"   class="flex justify-between border rounde
mb-5">

                                    <span wire:sortable.handle class="grid place-content-cente
bg-stone-50 hover:bg-sky-50 px-3 cursor-move ">
                                        <i class="fa-solid fa-grip-vertical "></i>
                                    </span>
                                    <span  class=" w-full p-2 flex justify-between py-5">
                                        {{ $softwareSubcategoria->nombre }}
                                    </span>

                                </li>
                        @endforeach
                    </ul>
                </div>
            </div>
        </x-slot>

        <x-slot name="footer">
                                        <x-primary-button      type="button"      class="mr-3"
wire:click="cerrarModalReasignarOrdenTienda" target="guardar">
                Actualizar orden
            </x-primary-button>
                <x-secondary-button  type="button"  class=""  wire:click="$toggle('modalAbierto')"
wire:loading.attr="disabled">Cerrar</x-secondary-button>
```

6. Por último, se modificó el código fuente del botón “Reasignar orden en Portal de Software UNAM” de las categorías, definido en la vista Blade *listar-software-categorias.blade.php*:

https://doi.org/10.22201/dgtic.ctud.2025.3.1.100 **Vol. 3, Núm. 1.** enero-marzo 2025, págs. 50 - 50

Figura 14

Código fuente del botón "Reasignar orden en Portal de Software UNAM" en la vista listar-software-categorias.blade.php

```
<!-- resources/views/livewire/software-categorias/listar-software-categorias.blade.php -->

<div class="flex flex-row justify-center items-center">
    @can('registrar-software-subcategoria')
        <x-action-button class="bg-sky-600" title="Actualizar subcategoria de software"
data-tippy="Actualizar subcategoria de software"
wire:click="$dispatch('abrir-modal-registrar-software-subcategoria', { idSoftwareCategoria: {{
$softwareCategoria->id_software_categoria }}, idSoftwareSubcategoria: {{
$softwareSubcategoria->id_software_subcategoria }} })">
            <i class="fa-solid fa-pen-to-square"></i>
        </x-action-button>
    @endcan
</div>
```

El evento "clic" del botón, ejecuta *"abrir-modal-reasignar-orden-tienda-software-subcategoria"* para mostrar la ventana modal con el listado de Subcategorías de Software pertenecientes a la misma categoría. Al arrastrar y soltar algún elemento del listado, se llama al evento *actualizarOrdenTiendaSoftwareSubcategorias,* que realiza la actualización en la base de datos de manera dinámica.

De manera general, la estructura de directorios y archivos de PHP necesarios son:

Figura 15

Estructura de directorios y archivos Módulo de Categorías y Subcategorías de Software Laravel 11 y Livewire 3

```
├── app
│   ├── Livewire
│   │   ├── Forms
│   │   ├── SoftwareCategorias
│   │   │   ├── ReasignarOrdenTiendaSoftwareCategoriasComponent.php
│   │   ├── SoftwareSubcategorias
│   │   │   ├── ReasignarOrdenTiendaSoftwareSubcategoriasComponent.php
├── modulos
│   ├── SoftwareCategorias
│   │   ├── Actions
│   │   │   ├── ReasignarOrdenTiendaSoftwareCategoriasAction.php
│   │   ├── Enums
│   │   ├── Filters
│   │   ├── Models
│   │   │   └── SoftwareCategoria.php
│   │   └── QueryBuilders
│   ├── SoftwareSubcategorias
│   │   ├── Actions
│   │   │   ├── ReasignarOrdenTiendaSoftwareSubcategoriasAction.php
│   │   ├── Enums
│   │   ├── Filters
│   │   ├── Models
│   │   │   └── SoftwareSubcategoria.php
│   │   └── QueryBuilders
├── resources
│   └── views
│       ├── livewire
│       │   ├── software-categorias
│       │   │   ├── reasignar-orden-tienda-software-categorias.blade.php
│       │   ├── software-subcategorias
│       │   │   ├── reasignar-orden-tienda-software-subcategorias.blade.php
```