

BSale Test Tienda

Test postulación: cargo Desarrollador en Bsale.

Nombre: Franco Bernal Gutiérrez.

Título: Análisis y programación de software.

Repositorio: <https://github.com/franco-bernal/BsaleTest>

Web test: <https://devfranco.tk/>

La api ahora se encuentra en <https://devfrancoback-end.tk/>

Las url disponibles son

<https://devfrancoback-end.tk/api/category>

<https://devfrancoback-end.tk/api/product>

<https://devfrancoback-end.tk/api/bycategory/name/category/orden>

(pasándole datos a la última en name, category y orden)

La aplicación fue desarrollada con PHP en el framework Laravel 8, utilizando además jQuery y Bulma.io como apoyo. La aplicación está en un VPS, específicamente en hosty.cl, usando Linux Ubuntu 18.04 Server X86 64 Min Gen2 V1 y se usó además GitHub para el versionamiento.

Para este ejercicio se crearon 3 Apis, llamadas category, producto y bycategory. La primera entrega información sólo de las categorías; la segunda sobre todos los productos y la tercera es la más importante, se encarga de filtrar los datos. Para esto bycategory recibe en su ruta 3 parametros, name, category y orden, los cuales se procesan de acuerdo con su contenido dependiendo estén llenos o vacíos y devolviendo el resultado de una consulta, las que pueden ser buscar el nombre, y/o la categoría y además devolverlas ordenadas por el nombre desde la A a la Z o viceversa o por precio desde el menos costoso hasta el más costoso.

Imagen del método que corresponde a la api bycategory:

```
//Metodo que procesa los filtros de nombre, categoría y orden
public function productByCategory($name = "all", $category = "all", $orden = "asc", Request $request)
{
    $name = strtolower(request('name'));
    $category = request('category');

    if ($name == "all") {
        $name = "";
    }
    if ($category == "all") {
        $category = "";
    }

    $products = Product::where('category', 'LIKE', "%".$category."%")
        ->where('name', 'LIKE', "%".$name."%")
        ->with('category');

    if ($orden == "mas") {
        return $products->orderBy('price', "desc")->paginate(7);
    }
    if ($orden == "menos") {
        return $products->orderBy('price', "asc")->paginate(7);
    }

    return $products->orderBy('category', $orden)->paginate(7);
}
```

Opcionalmente se puede ingresar a las api category y producto.

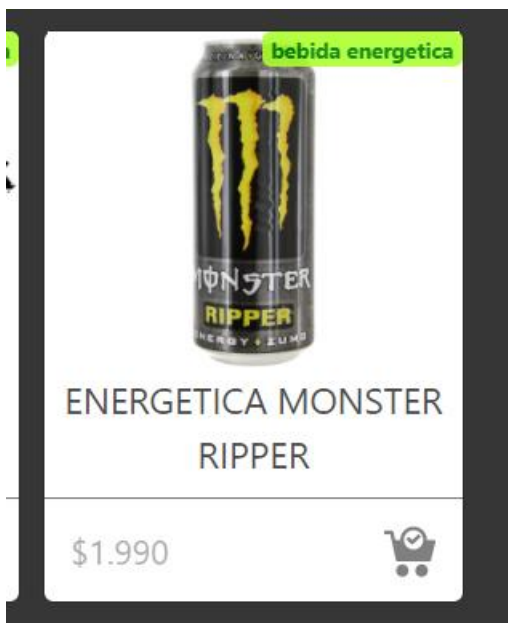
```
url: "https://devfrancoback-end.tk/api/category",
url: "https://devfrancoback-end.tk/api/product",
```

Para implementar esto en el cliente se usó Ajax con ayuda de jQuery, el método principal es loadTable() que se encarga de ir a buscar los datos y cargar a loadProducts() su data para poder dibujarlo en el html.

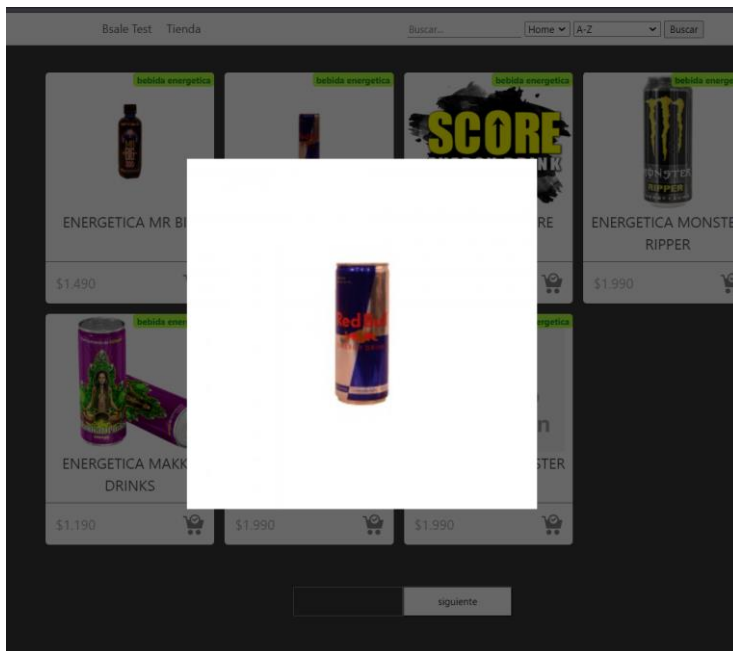
```
... function loadTable(filtro: any, page: any): void { //ckend para traer los productos
... function loadTable(filtro, page) {
...     url_current = filtro;
...     current_page = page;
...     $("#bs-count").css('display', 'none');
...     $('.product-space').html('<div class="loader is-loading centrar"></div>');
...     $('#pagination').html('');
...
...     $.ajax({
...         type: 'GET',
...         data: {
...             page: page,
...         },
...         url: "https://devfranco.tk/api/" + filtro,
...         success: function(data) {
...
...             $(".header").slideDown();
...
...             $('#pagination').html("");
...             if (data['data'] == "") {
...                 $('.product-space').html(`
...                 <h1 style="text-align: center; font-size:20px;color:white;">Sin coincidencias</h1>`);
...             } else {
...                 loadProducts(data);
...             }
...         },
...         error: function(error) {
...             console.log(error);
...         }
...     })
... }
```

Esta función necesita 2 argumentos, el filtro, que es un string con la url que menciona la api a utilizar, siendo por ejemplo 'bycategory/1/3/asc' siendo igual al modelo de la api 'bycategory{name}/category/orden'. Y además necesita que le indiquen la página, para poder devolver los datos paginados desde el servidor, precaviendo así que no se muestren demasiados productos y puedo optimizar los tiempos de respuesta en la aplicación.

Para el usuario final se agregaron algunos detalles que sea más intuitivo, por ejemplo se agregaron tags en cada producto indicando la categoría a la que pertenecen, esto además, si se le hace CLIC, Podrá ver todos los resultados que tengan la misma categoría.



Además usualmente el usuario final sabe que toda app puede ver de cerca el producto que quiere y es indispensable, por lo que se le agregó un modal con la imagen en un tamaño más grande para este fin, lo que es beneficioso para productos que no se vean bien definidos o se vean muy pequeños. Esto sólo haciendo clic en la imagen del producto que desea visualizar.



Se validó que los productos tuvieran imagen, en caso de tener, se le agrega una por default indicando que no está disponible.

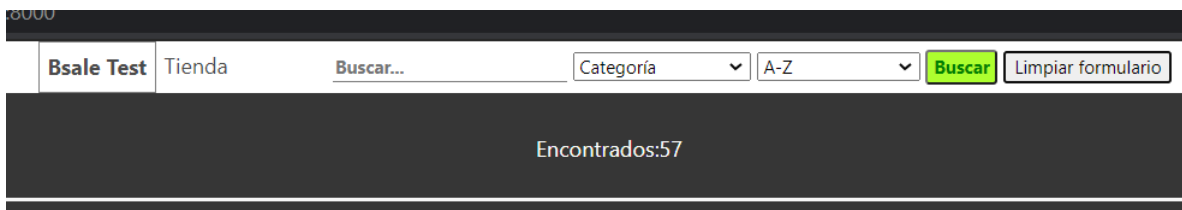
```
//valida si existe imagen, si no entrega una por default
function imageValid(image) {
  if (image == "" || image == null) {
    return "https://devfranco.tk/test/img/default/product_error.jpg";
  }
  return image;
}
```

También se agregó una función para agregar los miles a los precios.

La función convierte a String el precio lo separa cada 3, agrega los puntos y luego junta toda la cadena.

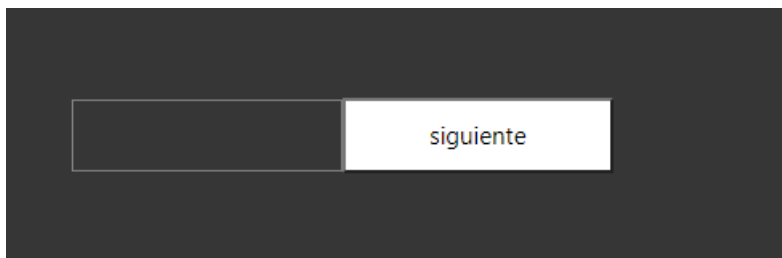
```
//función que agrega los puntos a los precios
function numberFormat(value) {
  if (integerValid(value)) {
    var retorno = '';
    value = value.toString().split('').reverse().join('');
    var i = value.length;
    while (i > 0) retorno += ((i % 3 === 0 && i !== value.length) ? '.' : '') + value.substring(i--, i);
    return retorno;
  }
  return 0;
}
```

Otras funciones:



A screenshot of a search interface. At the top left, there is a dark button labeled 'Bsale Test'. To its right is the word 'Tienda'. Further right is a search input field with the placeholder text 'Buscar...'. To the right of the input field is a dropdown menu labeled 'Categoría' with a downward arrow, and another dropdown menu labeled 'A-Z' with a downward arrow. To the right of these is a green button labeled 'Buscar' and a light gray button labeled 'Limpiar formulario'. Below this row is a dark horizontal bar with the text 'Encontrados:57' in the center.

- Al hacer click en Bsale Test puede volver a los resultados principales.
- Puede limpiar el formulario con un botón.
- Se muestran la cantidad de datos encontrados.
- Los botones de la paginación se deshabilitan cuando llegan a su límite de páginas



A screenshot of a pagination control. It consists of a dark rectangular button on the left and a light gray rectangular button on the right. The light gray button contains the text 'siguiente'.

Cabe mencionar que también se agrego responsive por lo que también se puede visualizar desde cualquier dispositivo.