



FASTPART

Proyecto Integrado DAW



FastPart

2023-2024

ISAAC DELGADO ROMÁN

Contenido

Presentación del proyecto	2
Funcionalidades de la página.	3
Tecnologías utilizadas.....	4
Tecnologías de desarrollo.....	4
Entornos de desarrollo	8
Herramientas de diseño	9
Guía de estilos.....	10
Paleta de colores	10
Fuentes	10
Iconos	10
Diseño base de datos	20
users	20
categories.....	21
orders.....	21
order_product.....	21
Implementación del proyecto	22
Backend	22
Frontend.....	25
Conclusiones	29
Dificultades encontradas	29
Conocimientos adquiridos	30
Bibliografía:	30



Presentación del proyecto

¿Eres un pequeño emprendedor que acaba de heredar o crear un taller y tienes muchas partes a las que no puedes darle salida? Fastpart es tu solución.

¿Tienes varios coches y quieres vender sus partes? Fastpart es tu solución.

¿Necesitas comprar una parte de algún coche a un precio barato y en una comunidad de vendedores dedicados? Fastpart es tu solución.

Fastpart es una página de compra/venta de partes de coche, en la cual habrá usuarios que puedan vender, estos tendrán que pedirle a los administradores autorización. También habrá los usuarios compradores, los cuales no tendrán que hacer nada más que crear la cuenta.



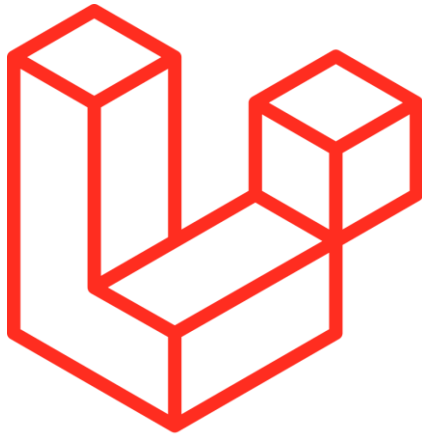
Funcionalidades de la página.

- Identificación con Google.
- Verificación de cuenta mediante correo electrónico.
- Compra y venta de partes de coches.
- Carrito de compra dinámico.
- Perfil de usuario.
- Filtrado de productos por precio, marca y categoría.
- Creación de productos por parte del admin y de los vendedores.
- Gestión de usuarios.



Tecnologías utilizadas

Tecnologías de desarrollo



Laravel, un framework del lenguaje de programación PHP, el cual otorga facilidades como comunicación directa con la base de datos sin tener que hacer peticiones a una API, la opción de poder verificar la cuenta mediante el envío de correos electrónicos sin tener que hacer nada, da facilidades a la hora de seguir el modelo MVC (mediante creación dinámica de modelos, controladores y vistas) y creación de Middlewares. También otorga la creación de migraciones, para poder crear la base de datos con un solo comando.



LIVEWIRE

Livewire, otro framework de PHP, pero este para no tener que hacer peticiones al servidor y poder hacer la página más dinámica, con el fin de darle parecido a una página hecha con Angular, con lo positivo de esto. Se usa para el eliminado dinámico de productos del carro, actualización dinámica de precio total del carro y, toda la gestión de productos del admin.



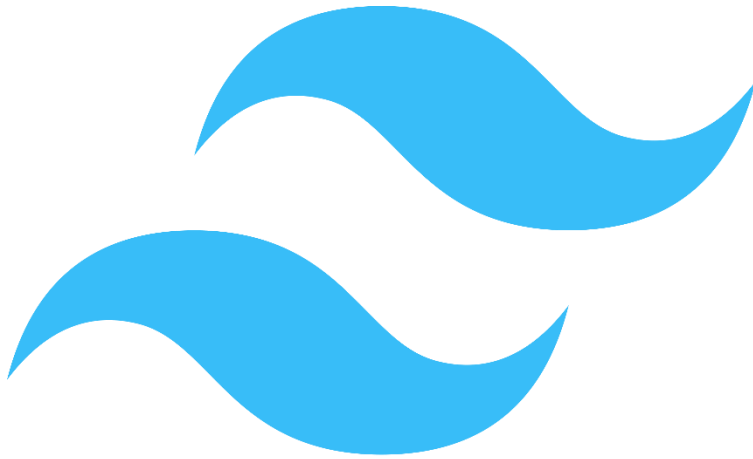


JavaScript, debido a que muchas de las cosas que usan tanto Livewire como Laravel se apoyan en JavaScript, pero yo no he programado casi nada usando este lenguaje.



MySQL como gestor de bases de datos, instalado en local, para tener la base de datos en mi pc y poder monitorizarla correctamente.





Tailwind como framework de CSS, para poder facilitar todo el diseño de la página y poder estructurarla más.

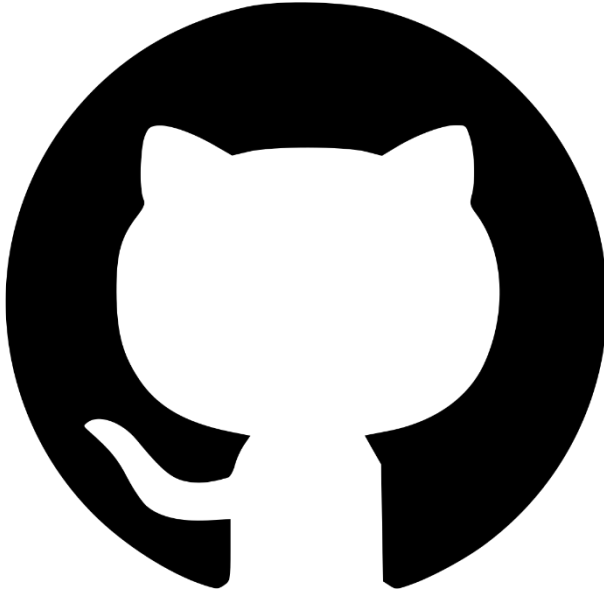


Flowbite, como una ampliación de Tailwind, en donde vienen bastantes componentes prehechos.



SCSS como framework de CSS para compilar este y poder escribir mucho menos código.





GitHub para el control de código, debido a que es gratuito, contiene bastantes funciones y además es muy sencillo e intuitivo en el uso.



Entornos de desarrollo



PHPStorm para todo el proyecto, debido a que es un IDE dedicado totalmente a PHP y a sus frameworks, también equipado con sistema de IA y con plugins que se pueden ir instalando, me ha dado muchas facilidades, más que Visual Studio Code.



DataGrip para la base de datos, tiene más opciones que MySQL Workbench y es más concreto, además de resultarme más fácil de entender.



Herramientas de diseño

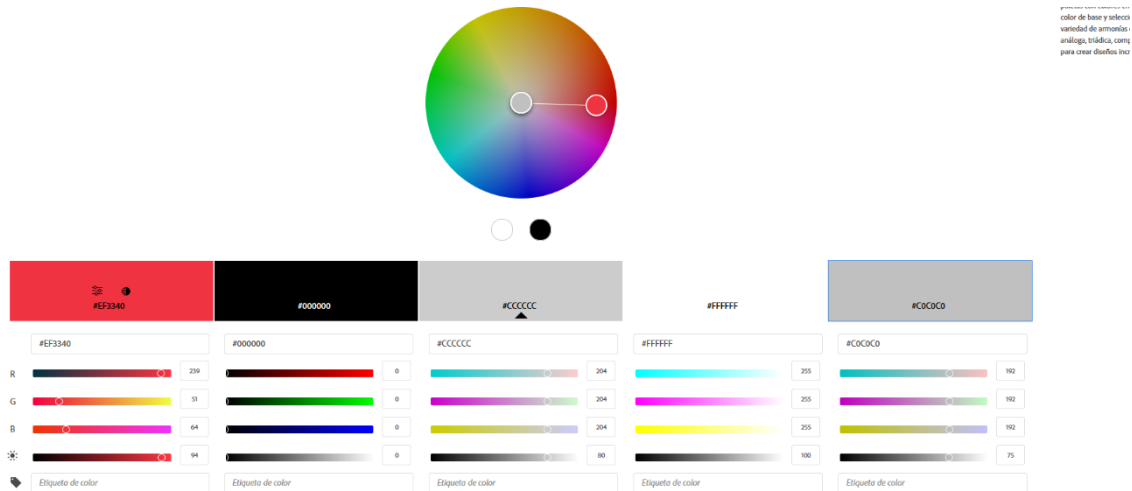


Figma, que me ha permitido una creación sencilla e intuitiva de una base para toda la interfaz de usuario.



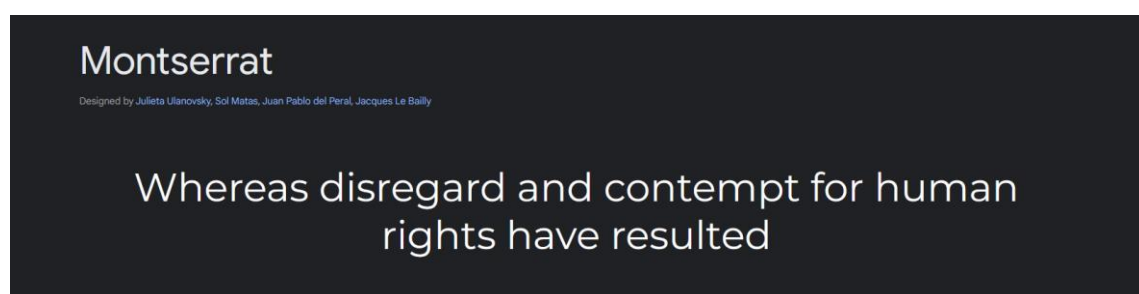
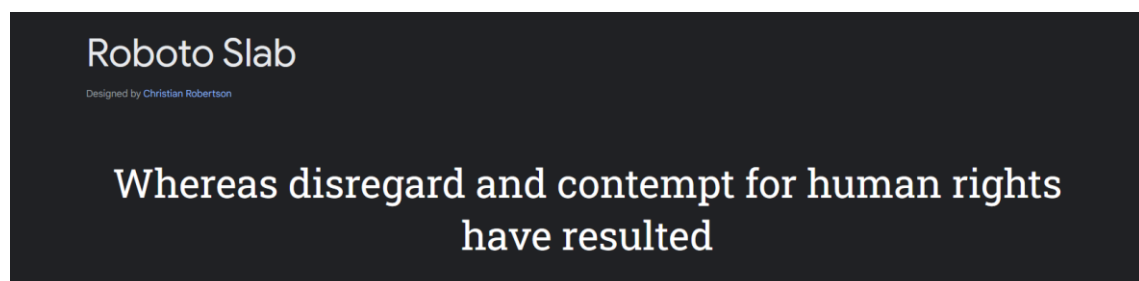
Guía de estilos

Paleta de colores



He elegido esta paleta de colores claros, para que sea agradable a la vista, con un rojo bastante más llamativo, para que cosas como precios o cosa importantes a la hora de mostrar un producto se vean de forma bastante más clara.

Fuentes



He usado estas dos fuentes para el proyecto.

Iconos



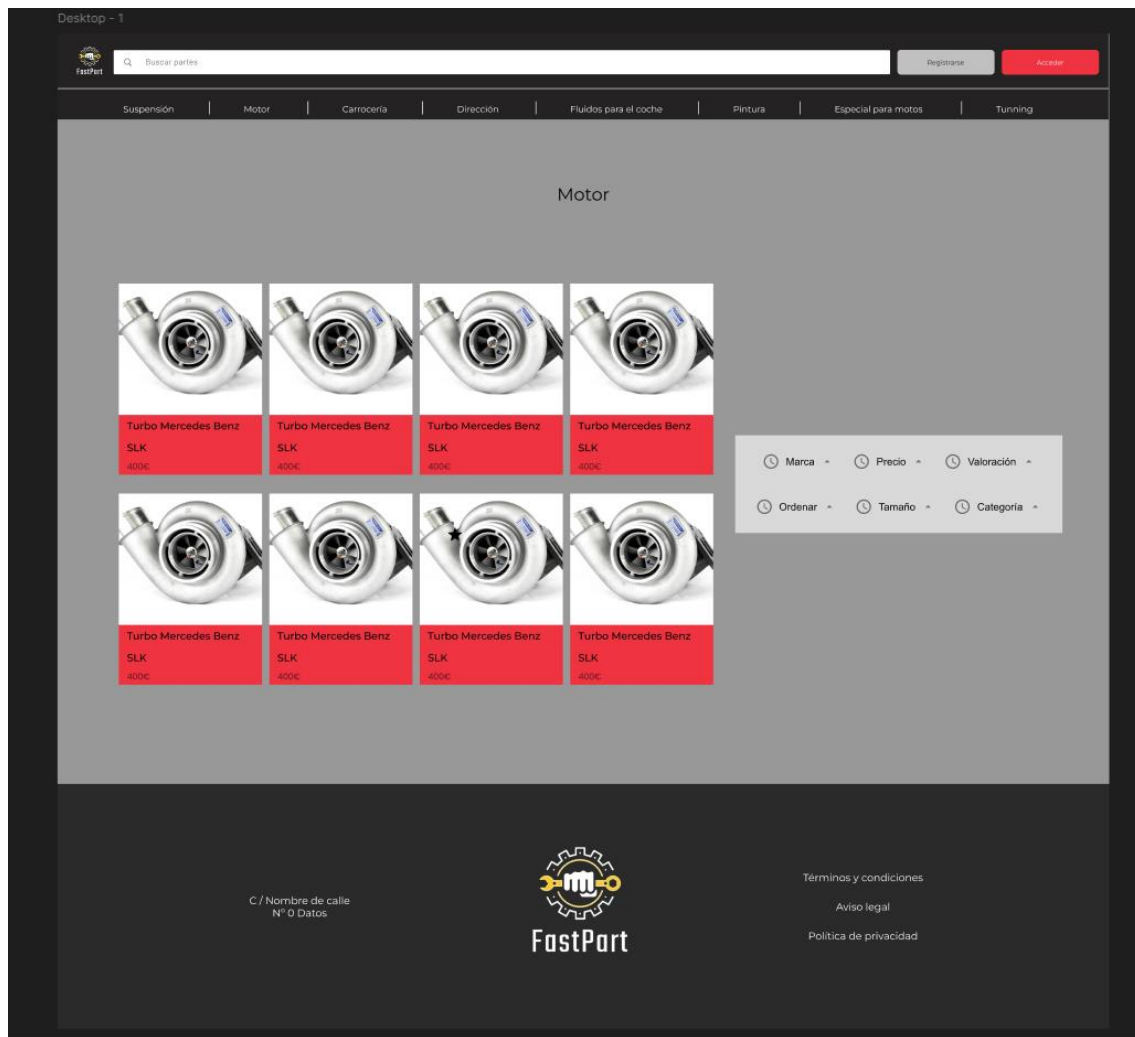



Para los iconos he usado Google Fonts.

Diseño de la página

Cabe agregar que esto fue un diseño preliminar, en el cual se han agregado y se han quitado diferentes funciones.








[Registrarse](#)
[Comprar](#)


[Suspensión](#)
[Motor](#)
[Carrocería](#)
[Dirección](#)
[Fluidos para el coche](#)
[Pintura](#)
[Especial para motos](#)
[Tuning](#)




¿QUIERES VENDER TAMBIÉN?

[CLICK AQUÍ PARA VENDER](#)


Ofertas de última hora.
¡Corre para aprovecharlas!



Turbo Mercedes Benz SLK
Turbo en perfecto estado, pocos meses de uso. Venta por necesidad.
400€ Isaac Delgado, hace 10 horas.




Turbo Mercedes Benz SLK
Turbo en perfecto estado, pocos meses de uso. Venta por necesidad.
400€ Isaac Delgado, hace 10 horas.



Turbo Mercedes Benz SLK
Turbo en perfecto estado, pocos meses de uso. Venta por necesidad.
400€ Isaac Delgado, hace 10 horas.


C / Nombre de calle
Nº 0 Datos



FastPart

[Términos y condiciones](#)
[Aviso legal](#)
[Política de privacidad](#)






[FastPart](#) [Iniciar sesión](#) [Registrarse](#)

[Suspensión](#) | [Motor](#) | [Carrocería](#) | [Dirección](#) | [Fluidos para el coche](#) | [Pintura](#) | [Especial para motos](#) | [Tuning](#)

Subir producto

Subir fotos del producto



[Añadir foto](#)

Nombre del producto

Marca del producto

¿Dónde se puede poner?

Coche del que se extrajo

Coches compatibles

Estado

Antigüedad


KM del producto

Precio del producto

Descripción del producto

[Subir producto](#)

C / Nombre de calle
Nº 0 Datos


FastPart

[Términos y condiciones](#)

[Aviso legal](#)

[Política de privacidad](#)




FastPart

Buscar partes

Registrarse Acceder

Suspensión Motor Carrocería Dirección Fluidos para el coche Pintura Especial para motos Tuning

Cambiar foto de perfil



Cambiar foto de perfil

Nombre de usuario:

Cambiar nombre de usuario

Correo electrónico:

Cambiar correo electrónico

Cambiar contraseña:


Contraseña actual:

Nueva contraseña:

Repetir contraseña:

Cambiar contraseña

C / Nombre de calle
Nº 0 Datos



FastPart

Términos y condiciones
Aviso legal
Política de privacidad



Desktop - 1

Registrarse

Nombre Contraseña

Correo electrónico

País Código postal

Contraseña

Confirmar contraseña

Teléfono

Fecha de nacimiento


☐ Soy mayor de 18

☐ Acepto los términos y condiciones

[Crear cuenta](#)

[Ya tengo cuenta](#)

C / Nombre de calle
Nº 0 Datos



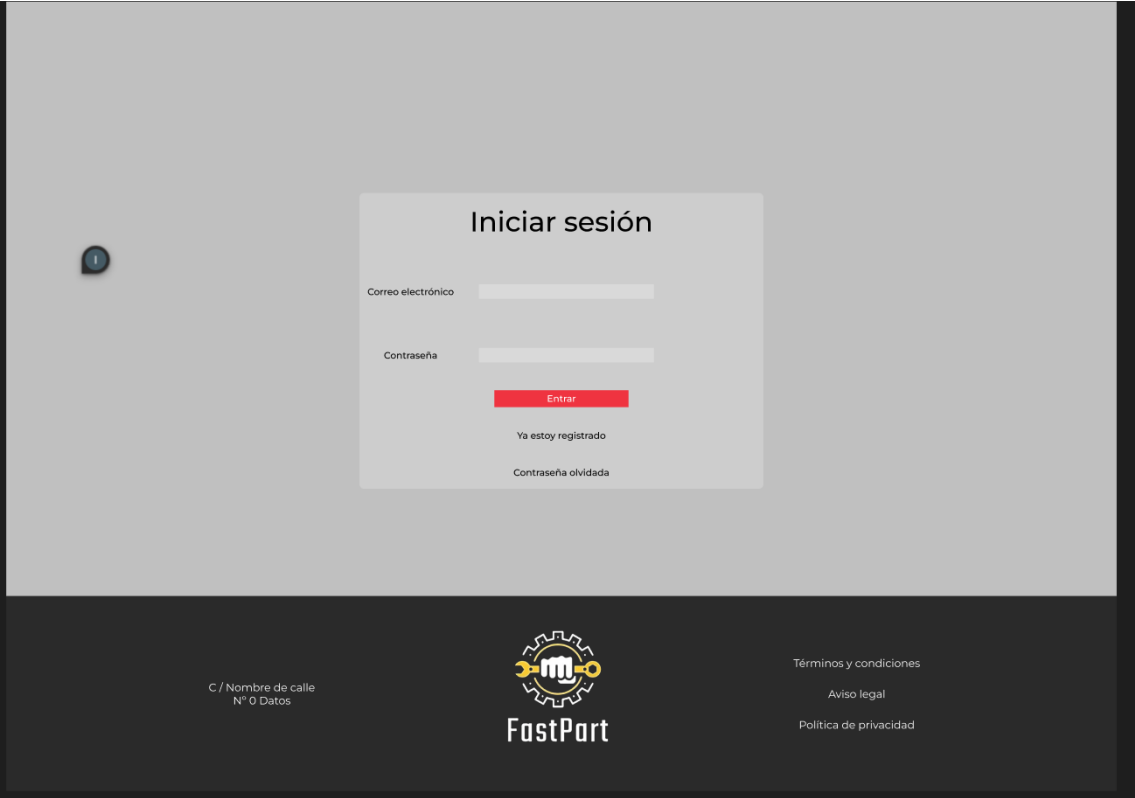
FastPart


[Términos y condiciones](#)

[Aviso legal](#)

[Política de privacidad](#)








Buscar partes


RegistrarseComprar


SuspensiónMotorCarroceríaDirecciónFluidos para el cochePneúmaEspecial para motosTuning

Turbo Mercedes Benz SLK




Isaac Delgado Román






400€


Turbo en perfecto estado, pocos meses de uso. Venta por necesidad.



Isaac Delgado Román


Muy buen vendedor, el producto me llegó pronto






Isaac Delgado Román


Muy buen vendedor, el producto me llegó pronto





Isaac Delgado Román

Muy buen vendedor, el producto me llegó pronto



C./ Nombre de calle
Nº 0 Datos



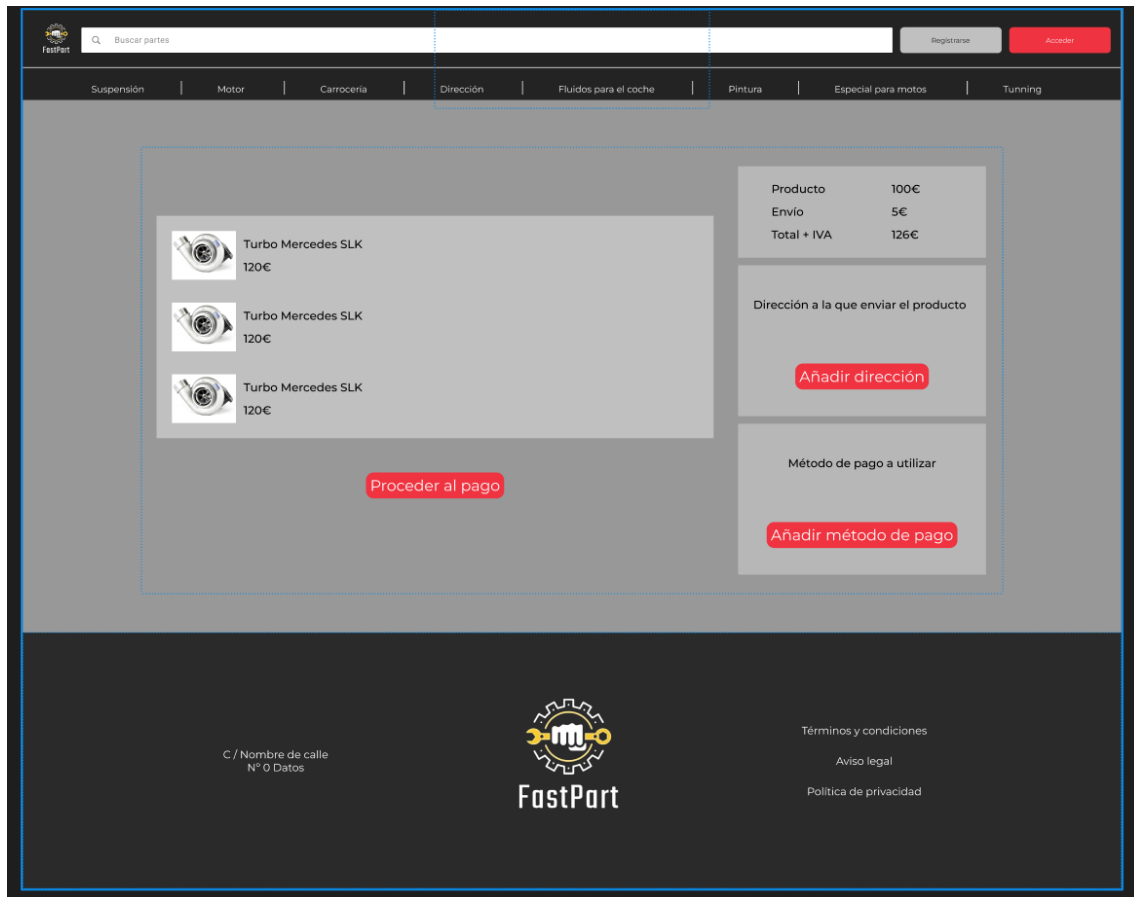
FastPort

Términos y condiciones

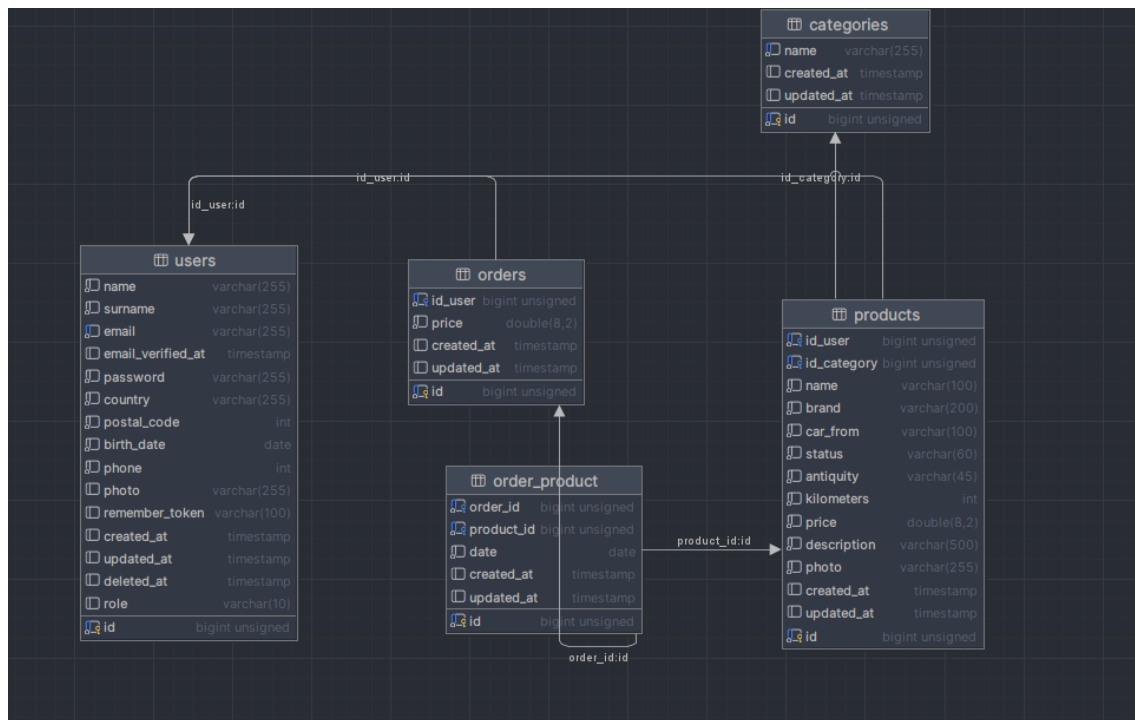
Aviso legal

Política de privacidad





Diseño base de datos



users

Id: Clave primaria

name: nombre del usuario.

surname: apellido del usuario.

email: email del usuario, debe de ser único.

email_verified_at: la fecha en la que se verificó la cuenta.

password: contraseña encriptada.

country: país del usuario.

postal_code: código postal del usuario.

birth_date: fecha de nacimiento.

phone: teléfono del usuario.

El resto de columnas están puestas por el framework, por lo que no las explicaré, ni en esa, ni en ninguna tabla.



categories

id: clave primaria.

name: nombre de la categoría.

products:

id: clave primaria.

id_category: id de la categoría, para relacionar las dos tablas.

name: nombre del producto.

brand: marca del producto.

car_from: coche del que viene la pieza.

status: estado de la pieza.

antiquity: antigüedad de la pieza, el año en el que fue hecha.

kilometers: kilómetros que tiene la pieza.

price: precio de la pieza.

description: descripción de la pieza.

photo: foto de la pieza.

orders

id: clave primaria.

id_user: id del usuario que ha puesto la orden.

price: dinero que ha costado la orden.

order_product

id: clave primaria.

order_id: identificador de la orden.

product_id: identificador del producto.

date: fecha en la que se ha puesto la orden.



Implementación del proyecto

Backend

La mayoría de cosas que hay que hacer para arrancar el proyecto se tienen que poner en el fichero .env, lo bueno de usar solamente Laravel es que la configuración es sencilla.

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=pruebafastpart
DB_USERNAME=root
DB_PASSWORD=root
```

Hay que buscar estos campos, para poner el tipo de base de datos, la IP, el puerto, el nombre de la base de datos y las credenciales para logearse.

También tendremos que poner las credenciales para hacer que el servidor de correo mande los correos al momento de que una persona se registre, en mi caso he usado el servidor de Google.

```
31 MAIL_MAILER=smtp
32 MAIL_HOST=smtp.gmail.com
33 MAIL_PORT=587
34 MAIL_USERNAME=isaacdelgado442@gmail.com
35 MAIL_PASSWORD="wgmf veej odef zngr"
36 MAIL_ENCRYPTION=null
37 MAIL_FROM_ADDRESS="isaacdelgado442@gmail.com"
38 MAIL_FROM_NAME="${APP_NAME}"
```

Tienes que poner esos datos.

Lo último que tenemos que hacer es poner las ids y las credenciales para configurar el login de Google que viene con el proyecto.

```
GOOGLE_CLIENT_ID=769970306077-j3udt05lucb5vahbh05e2b2r5qjjqtks.apps.googleusercontent.com
GOOGLE_CLIENT_SECRET=60CSPX-eAIh9nnR0p8VCp6ciVF7NxRl0gfs
GOOGLE_REDIRECT=http://127.0.0.1:8000/google/callback
```



Tras eso, abriremos una consola con la carpeta y para realizar las migraciones, simplemente tendremos que poner “php artisan migrate”

```
PS C:\Users\isaac\Desktop\fastpart> php artisan migrate
```

Esto hará las migraciones y creará la base de datos, para poder empezar a funcionar. También pondremos este comando si queremos llenar la base de datos con productos generados automáticamente por el propio Laravel:

```
PS C:\Users\isaac\Desktop\fastpart> php artisan db:seed
```

```
INFO Seeding database.
```

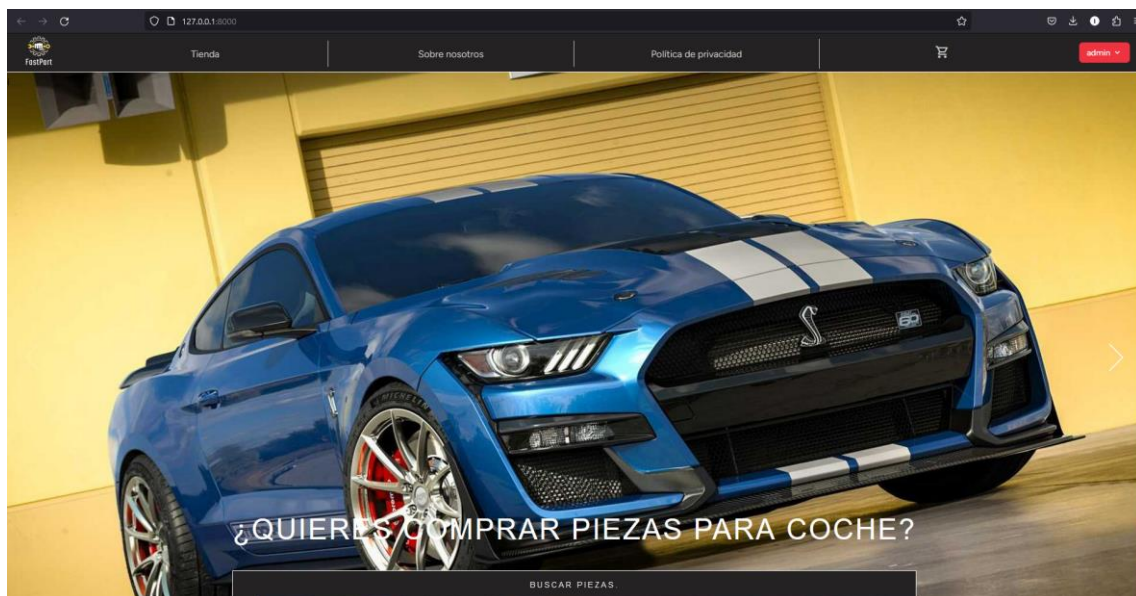
Finalmente, pondremos estos dos comandos para desplegar el proyecto en nuestro local host.

```
PS C:\Users\isaac\Desktop\fastpart> php artisan serve
```

Este para levantar el servidor.

```
PS C:\Users\isaac\Desktop\fastpart> npm run dev
```

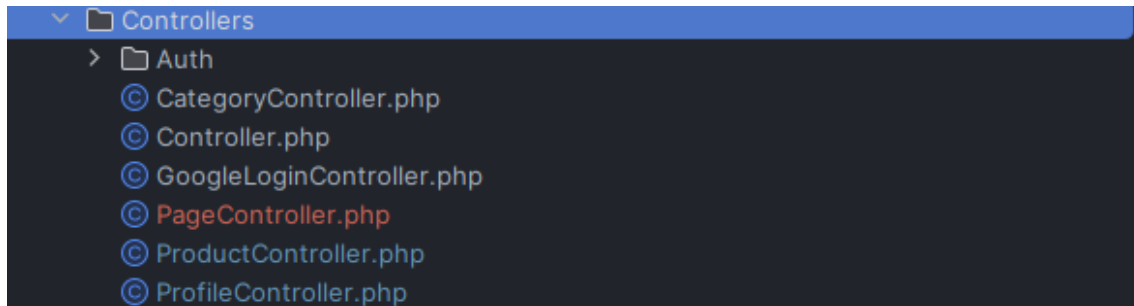
Este para iniciar el compilador de SASS.



Nos meteremos a la ip que nos da, 127.0.0.1:8000 y podremos ver nuestra página.



Para el Backend de la aplicación, como hemos usado el modelo MVC, he usado los controladores de los que te dota Laravel y que se pueden generar automáticamente.



Me ha hecho falta crear estos controladores para todo lo que necesito.

En los controladores, el acceso a la base de datos se hace mediante el ORM nativo de Laravel, es decir, Eloquent. Esto facilita mucho las consultas, además de ser bastante más atractivo a la vista y fácil de leer.

```

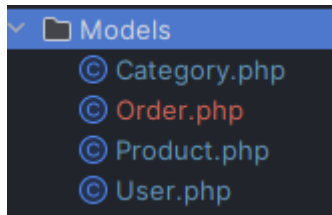
12 class ProductController extends Controller
13 {
14     /**
15      * Display a listing of the resource.
16      */
17     no usages new *
18     public function __construct()
19     {
20         $this->middleware(['auth', 'admin'])->only(['index']);
21         $this->middleware(['auth', 'seller'])->only(['store', 'create', 'update', 'mygames', 'edit', 'destroy']);
22     }
23
24     ↑ Isaac Delgado *
25     public function index()
26     {
27         return view( view: 'product.index');
28     }
29
30     1 usage ↑ Isaac Delgado *
31     public function myproducts()
32     {
33         $products = Product::where('id_user', Auth::user()->id)->get();
34         return view( view: 'product.myproducts')->with('products', $products);
35     }
36
37     /**
38      * Display the specified resource.
39      */
40     new *
41     public function show(Product $product)
42     {
43         return view( view: 'product.show')->with('product', $product);
44     }
45 }

```

Aquí hay un ejemplo de código.



Para el uso de estos controladores, necesitaremos generar los modelos de Laravel, tuve que generar estos.



```
↑ Isaac Delgado
public function user(): BelongsTo
{
    return $this->belongsTo(related: User::class, foreignKey: 'id_user', ownerKey: 'id');
}

new *
public function category(): BelongsTo
{
    return $this->belongsTo(related: Category::class, foreignKey: 'id_category', ownerKey: 'id');
}

no usages new *
public function orders()
{
    return $this->hasMany(related: Order::class);
}

}
```

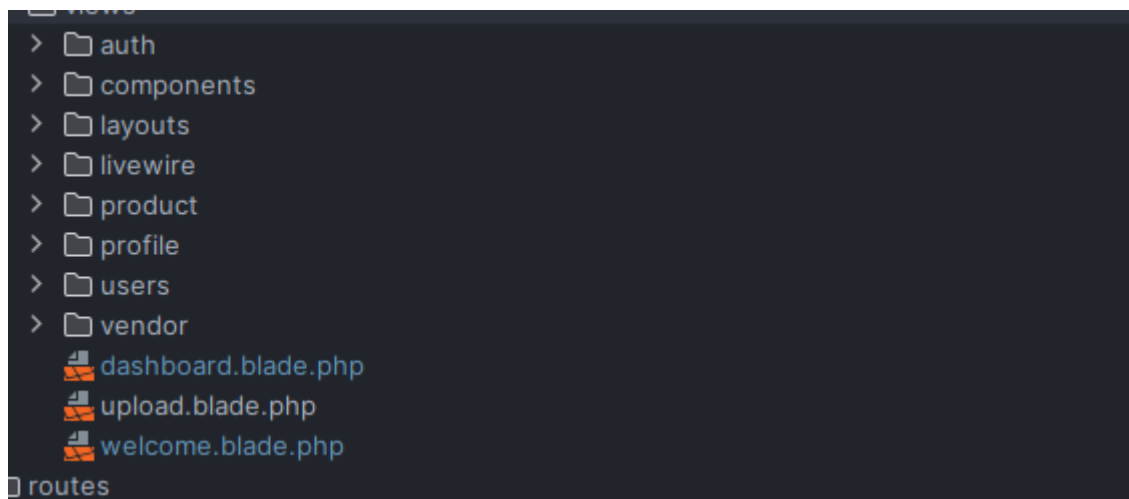
Aquí se usa también el ORM de Eloquent.

Frontend

Para el Frontend he usado los medios que me proporciona Laravel, lo primero es que para toda la gestión de usuario, he usado el framework de Breeze, que te hace automáticamente el login, el registro y para editar usuario.

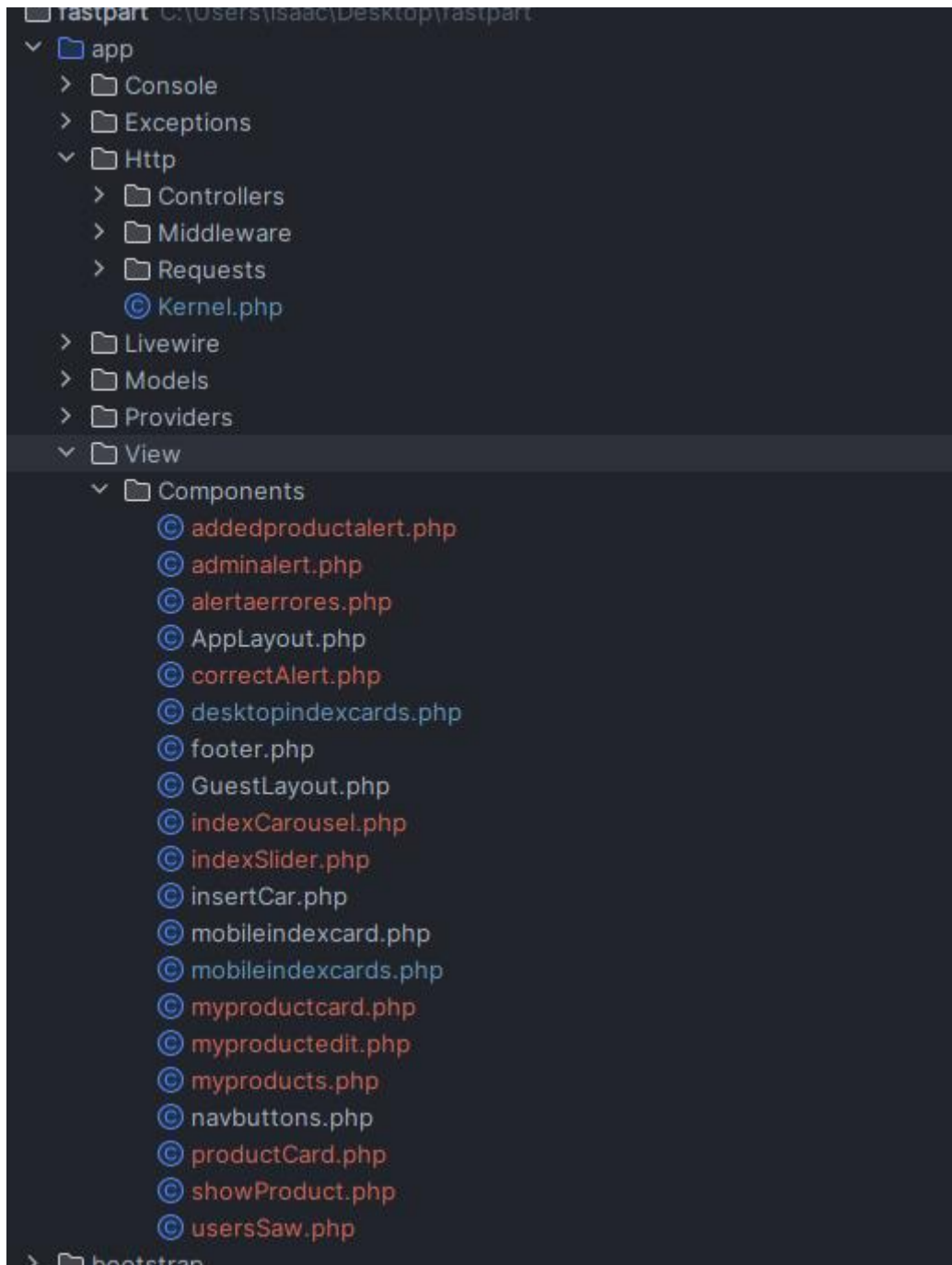
He basado el trabajo en componentes, tanto en componentes de Laravel como en componentes de Livewire.





Esos dos se separan en que los de Laravel necesitan hacer peticiones al servidor para recargar los datos y los de Livewire no. Estos componentes se pueden llamar en las vistas, tanto los de Laravel como los de Livewire.





Estos componentes también tienen una clase, para poder meterles atributos y poder utilizarlos mediante el motor de plantillas Blade.

Además de que he usado dos Middleware, uno para controlar las páginas a las que se pueden meter los admins y los vendedores.



```

class AdminMiddleware
{
    /**
     * Handle an incoming request.
     *
     * @param \Closure(\Illuminate\Http\Request): (\Symfony\Component\HttpFoundation\Response) $next
     */
    public function handle(Request $request, Closure $next): Response
    {
        if (Auth::user()->role == 'admin') {
            return $next($request);
        } else {
            return redirect()->route('route: 'dashboard')->with('message', 'No estás autorizado para acceder a esta página');
        }
    }
}

```

```

class SellerMiddleware
{
    /**
     * Handle an incoming request.
     *
     * @param \Closure(\Illuminate\Http\Request): (\Symfony\Component\HttpFoundation\Response) $next
     */
    public function handle(Request $request, Closure $next): Response
    {
        if (Auth::user()->role == 'seller') {
            return $next($request);
        } else {
            return redirect()->route('route: 'dashboard')->with('message', 'No tienes permisos para acceder a esta sección');
        }
    }
}

```

Cabe aclarar que Laravel tiene un sistema de routing mediante un archivo que se llama web.php, aquí dejo el sumario de las rutas que he usado:

```

Route::get(uri: '/', function () {
    return view('view: 'dashboard');
});

Route::get(uri: '/dashboard', function () {
    return view('view: 'dashboard');
})->middleware(['auth', 'verified'])->name(name: 'dashboard');

Route::middleware('middleware: 'auth')->group(function () {
    Route::get(uri: '/profile', [ProfileController::class, 'edit'])->name(name: 'profile.edit');
    Route::patch(uri: '/profile', [ProfileController::class, 'update'])->name(name: 'profile.update');
    Route::delete(uri: '/profile', [ProfileController::class, 'destroy'])->name(name: 'profile.destroy');
    Route::get(uri: '/uploadproduct', [App\Http\Controllers\ProductController::class, 'create'])->name(name: 'product.upload');
    Route::resource(name: '/product', controller: App\Http\Controllers\ProductController::class)->middleware(middleware: 'verified');
    Route::get(uri: '/allproducts', [App\Http\Controllers\ProductController::class, 'index'])->name(name: 'seeall');
    Route::get(uri: '/myproducts', [App\Http\Controllers\ProductController::class, 'myproducts'])->name(name: 'product.myproducts');
    Route::get(uri: '/cart', [App\Http\Controllers\ProductController::class, 'cart'])->name(name: 'cart');
    Route::post(uri: '/add-to-cart', [App\Http\Controllers\ProductController::class, 'addToCart'])->name(name: 'cart.add');
});

Route::middleware(middleware: 'verified');
Route::get(uri: '/google/redirect', [App\Http\Controllers\GoogleLoginController::class, 'redirectToGoogle'])->name(name: 'google.redirect');
Route::get(uri: '/google/callback', [App\Http\Controllers\GoogleLoginController::class, 'handleGoogleCallback'])->name(name: 'google.callback');
Route::controller(controller: TwitterController::class)->group(function () {
    Route::get(uri: 'auth/twitter', action: 'redirectToTwitter')->name(name: 'auth.twitter');
    Route::get(uri: 'auth/twitter/callback', action: 'handleTwitterCallback');
    Route::get(uri: '/users', [App\Http\Controllers\ProfileController::class, 'index'])->middleware(['auth', 'admin'])->name(name: 'users');
    Route::get(uri: '/productsindexadmin', [App\Http\Controllers\ProductController::class, 'adminGames'])->middleware(['auth', 'admin'])->name(name: 'products.index.admin');
});

Route::get(uri: '/about-us', [PageController::class, 'aboutUs'])->name(name: 'about-us');
Route::get(uri: '/cookies', [PageController::class, 'cookies'])->name(name: 'cookies');

```



Conclusiones

Dificultades encontradas

Entre ellas, varias, la primera es la de tener que configurar la librería de Socialite, para poder activar el login de Google, fue complicado de primeras, porque no entendí bien la documentación y no sabía bien a qué se refería, pero finalmente, pude sacarlo por mí mismo.

También la creación del carrito de la compra, había uno ya prehecho, pero quería hacerlo yo por mi cuenta para poder aprender, no sabía como podía hacerlo para eliminar los productos dinámicamente, pero al final, pude conseguirlo.

```

public function pay() {
    $order = new Order();
    $order->id_user = Auth::id();
    $order->price = $this->price;
    $order->save();
    foreach ($this->products as $product) {
        $order->products()->attach($product->id, ['date' => now()]);
    }
    Session::forget( keys: 'cart');
    $this->products = [];
    $this->price = 0;
    redirect( to: '/dashboard')->with('correct', 'Orden puesta correctamente, gracias por confiar en nosotros!);
}

```

Otra complicación es la gestión de fotos y de archivos con Laravel, muchas veces me daba errores que no sabía de dónde venían, al igual que los componentes de Livewire, pero esto es porque a la hora de poner muchas veces el mismo componente en un bucle, tienes que especificar un “id”, por así decirlo para que Livewire pueda gestionarlo bien, si no, da fallo.

```

@foreach($products as $product)
    <div class="col-span-1" wire:key="{{ Str::uuid() }}">
        <livewire:product-card-shop :$product wire:key="{{ Str::uuid() }}"></livewire:product-card-shop>
    </div>
@endforeach

```

Finalmente, la última complicación que tuve fue a la hora de los filtros, porque no entendía bien como se hacía, además de que se me juntó el mismo problema que he explicado con lo de los componentes en bucle, pero finalmente pude realizar todo.



Conocimientos adquiridos

- Implementación de Socialite con Laravel.
- Implementación de componentes de Livewire y profundización en todos estos.
- Creación de filtros en una página web (que me pareció lo más complicado de toda la página).
- Creación de Middlewares.
- Optimización de migraciones.
- Uso de nuevos IDEs de JetBrains como pueden serlo PHPStorm y DataGrip.
- Uso de la librería de “Flickity” para todos los carruseles de la página.
- Uso de Tailwind, estaba acostumbrado a Bootstrap.

Posibles ampliaciones

- Tramitar pagos con una plataforma como Paypal o Stripe.
- Poder admitir reembolsos y crear un historial de compras del usuario.
- Añadir valoraciones a cada producto y/o a cada vendedor.
- Introducir comentarios en cada producto.
- Permitir chat entre compradores y vendedores.

Bibliografía:

- Docs de [Tailwind](#)
- Docs de [Laravel](#)
- Docs de [Flowbite](#)
- Docs de [Flickity](#)
- Docs de [Livewire](#)
- Docs de [Figma](#)
- [Creador de logos](#)
- [Codepen](#)
- [ChatGPT](#)

