

Fastpart

Proyecto Integrado DAW



2023-2024

iSAAC dELGADO rOMÁN

Logotipo, nombre de la empresa

Descripción generada automáticamente

Contenido

[Presentación del proyecto 2](#_Toc169043629)

[Funcionalidades de la página. 3](#_Toc169043630)

[Tecnologías utilizadas 4](#_Toc169043631)

[Tecnologías de desarrollo 4](#_Toc169043632)

[Entornos de desarrollo 8](#_Toc169043633)

[Herramientas de diseño 9](#_Toc169043634)

[Guía de estilos 10](#_Toc169043635)

[Paleta de colores 10](#_Toc169043636)

[Fuentes 10](#_Toc169043637)

[Iconos 10](#_Toc169043638)

[Diseño base de datos 20](#_Toc169043639)

[users 20](#_Toc169043640)

[categories 21](#_Toc169043641)

[orders 21](#_Toc169043642)

[order\_product 21](#_Toc169043643)

[Implementación del proyecto 22](#_Toc169043644)

[Backend 22](#_Toc169043645)

[Frontend 25](#_Toc169043646)

[Conclusiones 29](#_Toc169043647)

[Dificultades encontradas 29](#_Toc169043648)

[Conocimientos adquiridos 30](#_Toc169043649)

[Bibliografía: 30](#_Toc169043650)

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.

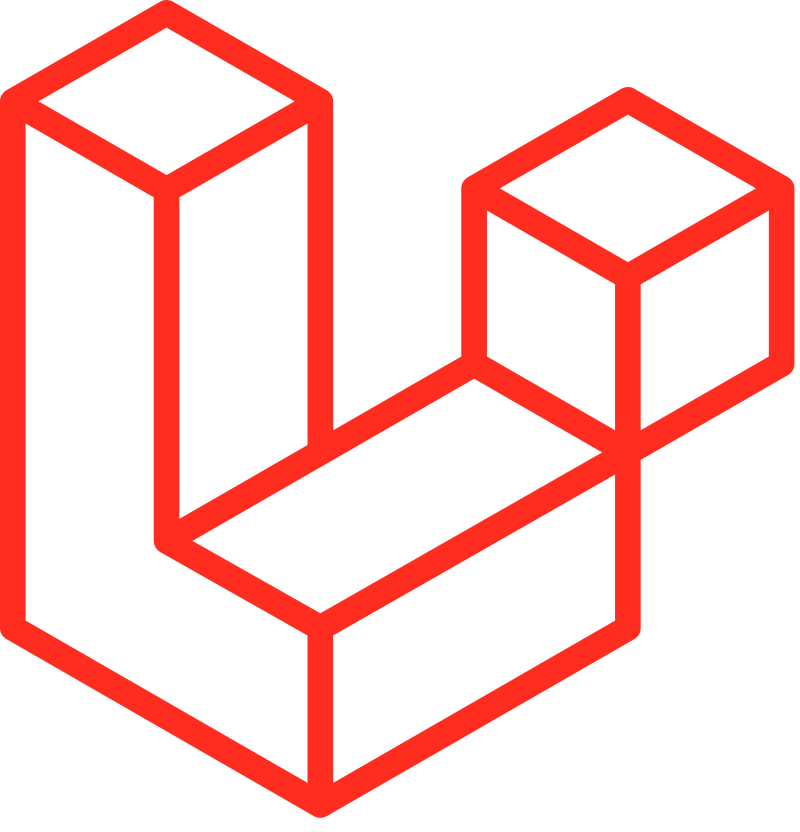
Fastpart es una página de compra/venta de partes de coche, en la cual habrá usuarios que puedan vender, estos tendrán que pedirle a los administradores autorización. También habrá los usuarios compradores, los cuales no tendrán que hacer nada más que crear la cuenta.

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

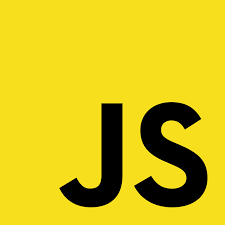
Tecnologías de desarrollo



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, 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.

Logotipo

Descripción generada automáticamente

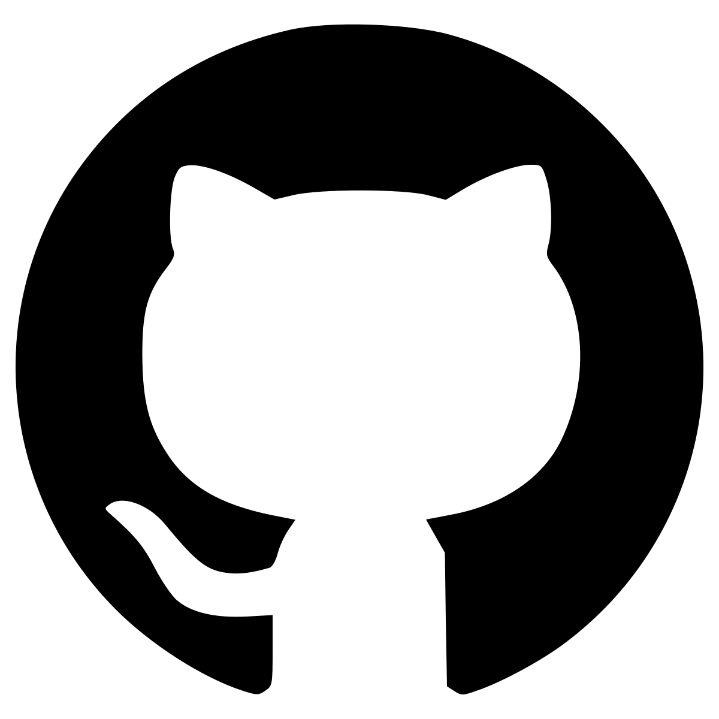
Tailwind como framework de CSS, para poder facilitar todo el diseño de la página y poder estructurarla más.



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

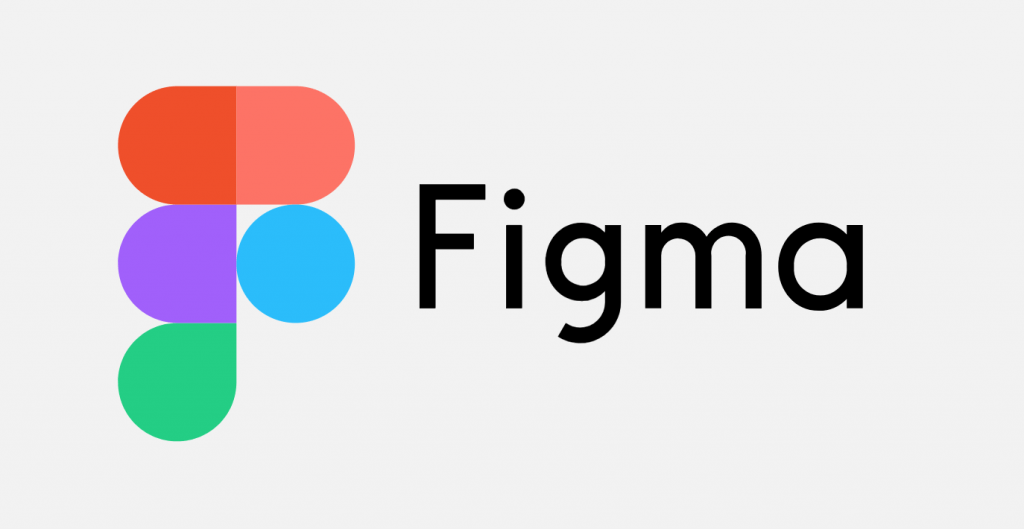


PHPStorm para todo el proyecto, debido a que es un IDE dedicado totalmente a PHP y a sus frameworks, también equipado con sistema de IA y con plugins que se pueden ir instalando, me ha dado muchas facilidades, más que Visual Studio Code.



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

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

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

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

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.

Interfaz de usuario gráfica

Descripción generada automáticamente

Página de búsqueda de la tienda, con filtros y con los productos.

Index de la tienda, en el que aparecen productos al azar y un carrusel.Interfaz de usuario gráfica, Sitio web

Descripción generada automáticamenteInterfaz de usuario gráfica, Aplicación

Descripción generada automáticamente Páginad e subida de producto apra los vendedores de la tienda.Interfaz de usuario gráfica, Sitio web

Descripción generada automáticamente

Página de gestión de perfil, para cambiar distintos atributos de tu cuenta.

Registro de la página.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Login de la página.Interfaz de usuario gráfica, Aplicación, Sitio web

Descripción generada automáticamente

Interfaz de usuario gráfica

Descripción generada automáticamente

Página de cada producto de la página.

Interfaz de usuario gráfica

Descripción generada automáticamente

Carrito de la compra.

Diseño base de datos

Imagen que contiene Texto

Descripción generada automáticamente

## users(usuarios de la página)

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 nacmineto.

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(Las distintas categorías que pueden tener los productos)

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(órdenes que los usuarios ponen al comprar)

id: clave primaria.

id\_user: id del usuario que ha puesto la orden.

price: dinero que ha costado la orden.

## order\_product(tabla para relacionar las órdenes con los productos, es una N:M)

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.

Texto

Descripción generada automáticamente

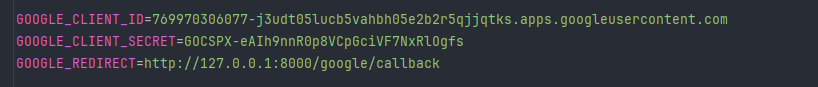
Hay que buscar estos campos, para poner el tipo de base de datos, la IP, el puerto, el nombre de la base de datos y las credenciales para logearse.

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. Texto

Descripción generada automáticamente

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.



Tras eso, abriremos una consola con la carpeta y para realizar las migraciones, simplemente tendremos que poner “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:

Texto

Descripción generada automáticamente

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



Este para levantar el servidor.



Este para iniciar el compilador de SASS.

Un coche deportivo de color azul

Descripción generada automáticamente

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.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada 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.

Texto

Descripción generada automáticamente

Aquí hay un ejemplo de código.

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

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

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.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

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.

Texto

Descripción generada automáticamente

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.

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

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:

Texto

Descripción generada automáticamente

Conclusiones

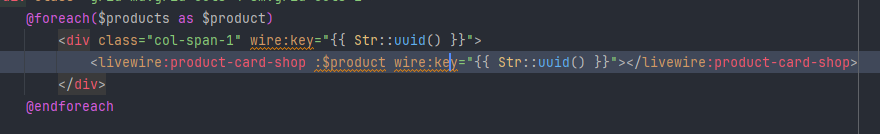
## Dificultades encontradas

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.



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.



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](https://tailwindcss.com/docs/utility-first)
* Docs de [Laravel](https://laravel.com/docs/10.x/installation)
* Docs de [Flowbite](https://flowbite.com/docs/getting-started/quickstart/)
* Docs de [Flickity](https://flickity.metafizzy.co/)
* Docs de [Livewire](https://livewire.laravel.com/docs/quickstart)
* Docs de [Figma](https://help.figma.com/hc/en-us)
* [Creador de logos](https://logo.com/)
* [Codepen](https://codepen.io/)
* [ChatGPT](https://chatgpt.com/)