HOJA DE RUTA - E-COMMERCE

Aquí tienes una guía básica para crear un ecommerce con carrito de compras utilizando Laravel. Esto te ayudará a llegar al Producto Mínimo Viable (PMV). Asegúrate de tener Laravel instalado antes de comenzar.

Paso 1: Configuración del Entorno

1.1 - DEPENDENCIAS COMPOSER Y NPM

Utiliza Composer para instalar Laravel:

`composer create-project --prefer-dist laravel/laravel nombre-del-proyecto`

Si composer ya está instalado actualizaremos composer cada vez que abramos nuestro proyecto:

`composer install`

`composer update`

Lo mismo con npm:

`npm install`

`npm update`

La opción --prefer-dist en el comando composer create-project indica a Composer que prefiera descargar e instalar distribuciones de paquetes comprimidas (archivos zip o tar) en lugar de clonar directamente desde el repositorio de versión de control (por ejemplo, GitHub).

Esta opción se utiliza para mejorar la velocidad de instalación, ya que las distribuciones comprimidas son más pequeñas y más rápidas de descargar que clonar el repositorio completo. Además, la opción --prefer-dist también evita la descarga de archivos innecesarios, como archivos de prueba y documentación, lo que puede ayudar a reducir el tamaño del proyecto resultante.

En el contexto de Laravel, esta opción se utiliza para crear un nuevo proyecto Laravel con la estructura de directorios y archivos necesarios sin tener que clonar todo el repositorio de Laravel desde GitHub, haciendo que el proceso de instalación sea más rápido y eficiente.

1.2 - ENV Y ENV.EXAMPLE

Configura tu archivo .env.example con los detalles de tu base de datos. Modificar las siguientes líneas:

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

Copiamos el archivo al .env:

```
cp .env.example .env
```

El comando cp .env.example .env se utiliza para copiar el contenido del archivo .env.example y crear un nuevo archivo llamado .env. En el contexto de Laravel, el archivo .env contiene la configuración específica de cada instalación, incluyendo detalles como la configuración de la base de datos, las credenciales de servicios, y otras variables de entorno.

En resumen, el comando cp .env.example .env es un paso común durante la configuración inicial de un proyecto Laravel, ya que establece el archivo de configuración personalizado que el framework utiliza para cargar la configuración específica de tu entorno.

1.3 - KEY GENERATE

Configurar la clave de la aplicación

```
php artisan key:generate
```

El comando php artisan key:generate en un proyecto Laravel sirve para generar una nueva clave de aplicación (app key) para tu aplicación. La clave de aplicación es una cadena alfanumérica que se utiliza para cifrar datos sensibles en tu aplicación, como las sesiones de usuario y las cookies.

Al ejecutar php artisan key:generate, Laravel generará automáticamente una nueva clave de aplicación y la colocará en tu archivo .env. Esto es especialmente útil cuando estás configurando un nuevo proyecto o cuando clonas un proyecto existente, ya que cada instalación debe tener su propia clave única y secreta.

Paso 2: Modelos y Migraciones

2.1 - MIGRATIONS

Laravel utiliza migraciones para gestionar la estructura de la base de datos. Puedes generar una migración con el siguiente comando:
`php artisan make:migration nombre_de_la_migracion`

Edita el archivo de migración generado en la carpeta database/migrations para definir la estructura de tus tablas. Ejecuta las migraciones para aplicar los cambios a la base de datos:
`php artisan migrate`

2.2 - MODELS

Los modelos en Laravel representan las tablas de la base de datos y te permiten interactuar con ellas de manera orientada a objetos. Genera un modelo con el siguiente comando:
`php artisan make:model NombreDelModelo`

Puedes especificar el nombre de la tabla a la que el modelo está asociado agregando la propiedad protected $table en el modelo.

Define las relaciones y otros métodos necesarios en el modelo.

2.3 - CONTROLLERS

Los controladores gestionan la lógica de la aplicación y la interacción con los modelos. Genera un controlador con el siguiente comando:
`php artisan make:controller NombreDelControlador`

Define los métodos en el controlador para realizar las acciones relacionadas con la base de datos, como mostrar registros, crear, actualizar o eliminar.

Paso 3: Seeders

3.1 - SEEDERS

Ejecutar seeders para llenar la base de datos con datos de prueba

```
php artisan db:seed
```

Ejemplo de un seeder o factory

```
'name' => $example->name,
```

Paso 4: Vistas y Rutas

4.1 - VISTA

Crea vistas para mostrar productos y el carrito de compras.

4.2 - RUTAS

Define rutas en routes/web.php para tus productos y carrito:

```
Route::resource('products', 'ProductController');
Route::get('cart', 'CartController@index');
Route::post('cart/add', 'CartController@add');
Route::post('cart/remove', 'CartController@remove');
```

Paso 5: Compilar recursos frontend

Compilar activos para desarrollo

```
npm run dev
```

6. Configuración de XAMPP

Iniciar XAMPP

Abre XAMPP y asegúrate de que Apache y MySQL estén iniciados.

Configurar Apache

Abre el archivo de configuración de Apache (ubicado en xampp/apache/conf/httpd.conf) y asegúrate de que el documento raíz apunte al directorio public de tu proyecto Laravel.

7. Configuración del Servidor de Desarrollo

```
Iniciar el servidor de desarrollo
php artisan serve
```