

# UD.1 Práctica Guiada: Configuración de un Entorno de Desarrollo Moderno con Docker y Laravel

---

(Contenidos de Guillermo Garrido adaptados por Eva María Gómez Abad- Curso 2025-2026)

## Objetivo

El objetivo de esta práctica es configurar un entorno de desarrollo profesional y desacoplado, utilizando herramientas modernas como un **entorno de línea de comandos potente, Docker y Laravel Sail**. Este enfoque aísla las dependencias de cada proyecto, evita conflictos en nuestro sistema anfitrión y nos prepara para un flujo de trabajo similar al que se encuentra en la industria del desarrollo de software.

## Filosofía de Trabajo

Antes de comenzar, es fundamental entender por qué adoptamos este modelo:

- **Sistema Anfitrión Limpio:** Mantenemos nuestro sistema operativo principal (Windows, macOS o Linux) limpio, utilizándolo para aplicaciones de interfaz gráfica como el navegador, Visual Studio Code y Docker Desktop.
  - **Entorno de Desarrollo Unificado:** Todas las herramientas de línea de comandos (CLI), lenguajes y servidores residirán en un entorno Unix-like (WSL/Ubuntu en Windows, o la terminal nativa en macOS y Linux). Esto nos brinda potencia y compatibilidad.
  - **Docker y Laravel Sail para Proyectos Aislados:** Cada proyecto se ejecutará en su propio conjunto de contenedores Docker. Esto garantiza que los proyectos no interfieran entre sí y que el entorno de desarrollo sea idéntico al de producción, eliminando el clásico problema de "en mi máquina funcionaba".
- 

## FASE 1: Cimentando las Bases (Terminal, Herramientas y Docker)

En esta fase instalaremos el software fundamental en nuestro sistema anfitrión.

### 1. Preparar el Entorno Base: Terminal y Gestor de Paquetes

Necesitamos una terminal potente y un gestor de paquetes para instalar software fácilmente.

== Windows

#### Instalar WSL2, Ubuntu 24.04 y Windows Terminal

Usaremos **PowerShell como Administrador** para estos comandos. WSL2 nos proporciona un entorno Linux completo dentro de Windows.

1. Instala WSL2 y la distribución de Ubuntu más reciente:

```
ws1 --install -d Ubuntu-24.04
```

El sistema se reiniciará. Al volver, se abrirá una terminal de Ubuntu pidiéndote que crees un **usuario** y una **contraseña**. ¡Anótalos!

2. Instala Windows Terminal para una mejor experiencia y **winget** para gestionar software:

```
winget install Microsoft.WindowsTerminal
```

Si **winget** no está disponible, descárgalo desde la [Microsoft Store](#).

== macOS

## Instalar Homebrew y Preparar la Terminal

Usaremos la **Terminal** nativa (o una alternativa como **iTerm2**) y el gestor de paquetes **Homebrew**.

1. Abre la aplicación **Terminal**.
2. Instala Homebrew (si no lo tienes) pegando el siguiente comando:

```
/bin/bash -c "$(curl -fsSL  
https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

Sigue las instrucciones que aparecerán en la terminal para añadir **brew** a tu PATH.

== Linux (Ubuntu)

## Preparar la Terminal y el Gestor de Paquetes

Usaremos la **Terminal** nativa y el gestor de paquetes **apt**.

1. Abre tu aplicación de Terminal.
2. Asegúrate de que tu sistema está actualizado (se te pedirá tu contraseña):

```
sudo apt update && sudo apt upgrade -y
```

## 2. Instalar Visual Studio Code y Docker Desktop

Estas son nuestras herramientas principales de desarrollo.

== Windows

## Instalar con Winget

Ejecuta estos comandos en **PowerShell** como **Administrador**:

```
winget install Microsoft.VisualStudioCode
```

Para Docker, descárgalo e instálalo desde su [página oficial](#). **Asegúrate de que la opción "Use WSL 2 based engine" esté marcada durante la instalación.**

== macOS

### Instalar con Homebrew

Ejecuta estos comandos en la **Terminal**:

```
brew install --cask visual-studio-code  
brew install --cask docker
```

Esto instalará VS Code y Docker Desktop. Inicia Docker Desktop desde tus aplicaciones.

== Linux (Ubuntu)

### Instalar con APT

1. **Visual Studio Code**: Descarga el paquete **.deb** desde la [página oficial](#) y haz doble clic para instalarlo, o sigue la guía oficial para añadir su repositorio.
2. **Docker**: Sigue la guía oficial para instalar **Docker Engine** y **Docker Compose**, ya que es el método recomendado para Linux: [Install Docker Engine on Ubuntu](#).

### 3. Configurar la Integración de Docker (Solo Windows)

Si estás en Windows, debemos asegurarnos de que Docker y WSL2 se comunican correctamente.

1. Abre **Docker Desktop**.
2. Ve a **Settings > Resources > WSL Integration**.
3. Asegúrate de que la integración con **Ubuntu-24.04** está **activada**.

---

## FASE 2: Preparando Nuestro Taller (Dentro del Entorno de Desarrollo)

[!IMPORTANT] ¡Atención!

A partir de ahora, todos los comandos se ejecutan dentro de tu entorno de desarrollo:

- **Windows**: Abre **Windows Terminal** o la **terminal de WSL** y selecciona el perfil de **Ubuntu**.
- **macOS/Linux**: Abre tu aplicación de **Terminal**.

### \*\*1. Instalar Herramientas Básicas de Desarrollo

Instalamos utilidades esenciales como **curl** y herramientas de compilación.

== Windows (WSL) / Linux (Ubuntu)

```
sudo apt update && sudo apt upgrade -y  
sudo apt install -y curl wget unzip build-essential
```

== macOS

```
brew install curl wget
```

## 2. Instalar Node.js a través de NVM (Node Version Manager)

NVM nos permite gestionar múltiples versiones de Node.js.

1. Ejecuta el script de instalación (universal para todos los SO):

```
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.40.0/install.sh |  
bash
```

2. **Cierra y vuelve a abrir la terminal** para que el comando `nvm` esté disponible.

3. Ahora, instala la última versión estable (LTS) de Node.js.

== Windows (WSL) / Linux (Ubuntu)

```
nvm install node # Instala la última versión
```

== macOS

```
nvm install --lts # Instala la última versión de soporte a largo plazo
```

4. Comprueba que todo está correcto (comando universal):

```
node -v && npm -v
```

## 4. Instalar PHP y Composer

Instalaremos PHP y su gestor de dependencias, Composer.

== Windows (WSL) / Linux (Ubuntu)

### Instalar con APT y PPA

Usaremos un repositorio externo de confianza para tener la última versión de PHP.

```
sudo add-apt-repository ppa:ondrej/php -y
sudo apt update
sudo apt install -y php8.4-cli php8.4-common php8.4-curl php8.4-mbstring php8.4-xml php8.4-zip
```

Ahora, instalamos Composer de forma global:

```
curl -sS https://getcomposer.org/installer | php
sudo mv composer.phar /usr/local/bin/composer
```

== macOS

### Instalar composer con Homebrew

Homebrew simplifica enormemente este proceso.

```
brew install php composer
```

Comprueba que las instalaciones son correctas (comandos universales):

```
php -v && composer -V
```

## 5. Instalar Extensiones en VS Code

== Windows

### Abrir VS Code desde WSL

1. Elige una de estas dos opciones:

- **Abre VS Code desde la terminal de WSL:**

```
code .
```

- **Abre VS Code desde VS Code directamente:** Es fundamental para conectar VS Code con tu entorno Ubuntu y ejecútalo bajo WSL.

2. **Ve al panel de Extensiones:** VS Code te pedirá instalar extensiones "en el servidor WSL". Instala las siguientes:

- Laravel Extra Intellisense (Amir)
- PHP Intelephense (Ben Mewburn)
- Laravel Blade Snippets (Winnie Lin)
- Tailwind CSS IntelliSense (Tailwind Labs)
- PHP Debug (xdebug)
- DBCode - Database Management (dbcode.io)

== Linux (Ubuntu) / macOS

## Abrir VS Code desde Terminal

### 1. Abre VS Code desde la terminal:

```
code .
```

### 2. Ve al panel de Extensiones e instala las siguientes:

- Laravel Extra Intellisense (Amir)
- PHP Intelephense (Ben Mewburn)
- Laravel Blade Snippets (Winnie Lin)
- Tailwind CSS IntelliSense (Tailwind Labs)
- PHP Debug (xdebug)
- DBCode - Database Management (dbcode.io)

## FASE 3: Creación y Vinculación del Proyecto "MyShop"

Esta fase es **universal** y se realiza completamente en tu terminal (Ubuntu en WSL, o la Terminal en macOS/Linux).

### 1. Crear el Directorio de Trabajo

```
cd ~  
mkdir -p development/laravel  
cd development/laravel
```

### 2. Crear el Nuevo Proyecto con Composer

```
composer create-project laravel/laravel MyShop  
cd MyShop
```

### 3. Abrir el Proyecto en VS Code

Ahora que ya tienes VS Code configurado, vamos a abrir la carpeta del proyecto que acabamos de crear.

== Windows

1. **Abre VS Code** (si no está ya abierto)
2. **Ve a File > Open Folder** (o presiona **Ctrl + K, Ctrl + O**)
3. **Navega a la ruta del proyecto:**

```
\\wsl$\\Ubuntu-24.04\\home\\[tu_usuario]\\development\\laravel\\MyShop
```

4. **Selecciona la carpeta MyShop** y haz clic en "Select Folder"
5. **Verifica que VS Code se conecta a WSL:** Deberías ver en la esquina inferior izquierda que está conectado a WSL

== macOS

1. **Abre VS Code** (si no está ya abierto)
2. **Ve a File > Open Folder** (o presiona **Cmd + O**)
3. **Navega a la ruta del proyecto:**

```
/Users/[tu_usuario]/development/laravel/MyShop
```

4. **Selecciona la carpeta MyShop** y haz clic en "Open"

== Linux

1. **Abre VS Code** (si no está ya abierto)
2. **Ve a File > Open Folder** (o presiona **Ctrl + K, Ctrl + O**)
3. **Navega a la ruta del proyecto:**

```
/home/[tu_usuario]/development/laravel/MyShop
```

4. **Selecciona la carpeta MyShop** y haz clic en "Open"

## FASE 4: ¡Zarpamos con Laravel Sail!

Ahora que el proyecto está creado vamos a darle vida con los contenedores de Sail. Todos los comandos se ejecutan desde la terminal, dentro del directorio de tu proyecto (**MyShop**).

### 1. Instalar y Configurar Sail

El comando **sail:install** preparará tu entorno Docker.

```
php artisan sail:install --with=mysql,redis
```

Cuando te pregunte qué servicios deseas, selecciona **mysql** y **redis** usando las flechas y la barra espaciadora.

## 2. Configurar el Archivo de Entorno (.env)

Asegúrate de que tu archivo **.env** está listo y contiene una APP\_KEY. De no ser así, genera lo y añade la clave de la aplicación.

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

```
php artisan key:generate
```

## 3. Crear un Alias para Sail

== Windows (WSL) / Linux (Ubuntu)

### Crear alias en bash

```
echo "alias sail='./vendor/bin/sail'" >> ~/.bashrc
```

```
source ~/.bashrc
```

== macOS

### Crear alias en zsh

```
echo "alias sail='./vendor/bin/sail'" >> ~/.zshrc
```

```
source ~/.zshrc
```

## 4. Levantar los Contenedores

Para que Docker aplique los cambios, ejecuta:

```
sail up -d --build
```

[!TIP] El flag `-d` (detached mode) El parámetro `-d` ejecuta los contenedores en segundo plano, liberando tu terminal para otros comandos. Sin este flag, la terminal quedaría ocupada mostrando los logs de los contenedores.

```
[+] Running 7/7
✓ sail-8.4/app          Built          0.0s
✓ Network myshop_sail   Creat...      0.0s
✓ Volume myshop_sail-redis Created        0.0s
✓ Volume myshop_sail-mysql Created        0.0s
✓ Container myshop-mysql-1 Started         0.8s
✓ Container myshop-redis-1 Started         0.8s
✓ Container myshop-laravel.test-1 Started        1.0s
```

[!CAUTION] Captura de pantalla requerida Realiza una captura de pantalla de tu terminal mostrando el comando `sail up -d --build` ejecutándose correctamente con todos los contenedores iniciados. La captura debe incluir tu prompt `usuario@equipo:~/ruta/proyecto$` visible.

[!WARNING] Problemas comunes Si tienes algun problema con el arranque de los contenedores, sigue los pasos de la sección de **Problemas de Arranque** para solucionarlo.

Si encuentras errores como "Another process with pid X is using unix socket file" o "Unable to setup unix socket lock file", sigue estos pasos:

#### 1. Detener todos los contenedores:

```
sail down -v
```

Este comando detiene y elimina únicamente los contenedores y volúmenes asociados a tu proyecto actual (MyShop), sin afectar a otros proyectos Docker que puedas tener en tu sistema.

#### 2. Limpiar volúmenes y contenedores:

```
# CUIDADO: Elimina contenedores detenidos y volúmenes no usados
docker system prune -f
```

Este comando elimina todos los contenedores detenidos, redes no utilizadas, imágenes colgantes y volúmenes no referenciados por ningún contenedor. Asegúrate de que no tienes datos importantes en volúmenes no referenciados antes de ejecutar este comando.

#### 3. Reiniciar Docker Desktop (Windows/macOS) o el servicio Docker (Linux):

- **Windows/macOS:** Cierra y vuelve a abrir Docker Desktop
- **Linux:** `sudo systemctl restart docker`

#### 4. Volver a levantar los contenedores:

```
sail up -d --build
```

Si tienes algun problema con los permisos de los archivos, sigue los pasos de la sección de **Problemas de Permisos en Linux (Ubuntu)** para solucionarlo.

Si después de ejecutar `sail up` te encuentras con errores de **"Permission Denied"** (Permiso denegado) al intentar ejecutar comandos o al cargar la web, es muy probable que los archivos de tu proyecto tengan permisos incorrectos.

Para solucionarlo, ejecuta estos comandos en la terminal de tu sistema (no dentro de Sail):

1. **Detener los contenedores:**

```
sail down
```

2. **Cambiar el propietario de los archivos** (sitúate en la carpeta de tu proyecto):

```
cd ~/development/laravel/MyShop  
sudo chown -R $USER:$USER .
```

3. **Volver a levantar los contenedores:**

```
sail up -d
```

Esto asegura que tu usuario sea el propietario de todos los archivos, evitando conflictos con Docker.

## FASE 5: Configuración de Bases de Datos

### 1. Preparar configuración en .env

Abre tu archivo `.env` y cambia las siguientes variables:

```
DB_CONNECTION=mysql  
DB_HOST=mysql  
DB_PORT=3306  
DB_DATABASE=myshop  
DB_USERNAME=sail  
DB_PASSWORD=password  
DB_EXTRA_OPTIONS=  
  
# Variable adicional para evitar warnings por consola  
MYSQL_EXTRA_OPTIONS=
```

## 2. Crear la Base de Datos

Antes de ejecutar las migraciones, debemos crear la base de datos **myshop** en MySQL.

Ejecuta el cliente de MySQL dentro del contenedor:

```
sail mysql -u root -ppassword
```

```
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A
```

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 20
Server version: 8.0.32 MySQL Community Server - GPL
```

```
Copyright (c) 2000, 2023, Oracle and/or its affiliates.
```

```
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.
```

```
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

```
mysql>
```

**Verifica con qué usuario estás conectado:**

```
SELECT USER();
```

```
+-----+
| USER()          |
+-----+
| root@localhost  |
+-----+
1 row in set (0.00 sec)
```

[!WARNING] ¿No estás conectado como root?

Si el resultado muestra **sail@localhost** en lugar de **root@localhost**,

Sal del cliente MySQL:

```
exit
```

Y conéctate usando **docker exec** directamente:

**1. Identifica el nombre de tu contenedor MySQL:**

```
docker ps --format "table {{.Names}}\t{{.Image}}" | grep -i mysql
```

Deberías ver algo como: **myshop-mysql-1** **mysql/mysql-server:8.0**

**2. Conéctate como root usando docker exec:**

```
docker exec -it myshop-mysql-1 mysql -u root -ppassword
```

**3. Verifica que estás conectado como root nuevamente:**

```
SELECT USER();
```

Una vez dentro del cliente MySQL **como root**, crea la base de datos **myshop**:

```
CREATE DATABASE IF NOT EXISTS myshop;
```

Query OK, 1 row affected (0.01 sec)

**Otorga permisos completos al usuario **sail** en la base de datos **myshop**:**

```
GRANT ALL PRIVILEGES ON myshop.* TO 'sail'@'%';
```

Query OK, 0 rows affected (0.01 sec)

**Aplica los cambios de permisos:**

```
FLUSH PRIVILEGES;
```

Query OK, 0 rows affected (0.00 sec)

Ahora verifica que se ha creado correctamente listando las bases de datos:

```
SHOW DATABASES;
```

```
+-----+
| Database          |
+-----+
| information_schema |
| laravel            |
| myshop             |
| mysql              |
| performance_schema |
| sys                |
+-----+
6 rows in set (0.00 sec)
```

Deberías ver **laravel** (creada por defecto por Sail) y **myshop** (la que acabas de crear).

[!NOTE] ¿Por qué la base de datos **laravel** ya existe? Al instalar Laravel Sail con MySQL, al iniciar el contenedor por primera vez, MySQL crea automáticamente una base de datos llamada **laravel**. ...

Para salir del cliente de MySQL, escribe:

```
exit
```

### 3. Ejecutar las Migraciones

Ahora que la base de datos **myshop** existe, podemos ejecutar las migraciones:

```
sail artisan migrate:refresh --seed
```

```
WARN  Migration table not found.

INFO  Preparing database.

Creating migration table ..... 72.26ms DONE

INFO  Running migrations.

0001_01_01_000000_create_users_table ..... 441.16ms DONE
0001_01_01_000001_create_cache_table ..... 125.26ms DONE
0001_01_01_000002_create_jobs_table ..... 306.37ms DONE
```

INFO Seeding database.

Si es la primera vez que ejecutas este comando, es posible que veas un mensaje indicando que no hay migraciones ejecutadas, o que veas las migraciones por defecto de Laravel (usuarios, tokens, etc.). Esto es normal y significa que la conexión funciona correctamente.

### 1. Configuramos la conexión con DBCode

Antes de configurar DBCode, necesitamos verificar el puerto exacto en el que MySQL está escuchando dentro de nuestro contenedor.

Ejecuta el siguiente comando para ver el estado de tus contenedores y sus puertos:

```
sail ps
```

NAME	IMAGE	COMMAND	
SERVICE	CREATED	STATUS	PORTS
myshop-laravel.test-1	sail-8.4/app	"start-container"	
laravel.test	7 minutes ago	Up 7 minutes	0.0.0.0:80->80/tcp, [::]:80->80/tcp, 0.0.0.0:5173->5173/tcp, [::]:5173->5173/tcp
myshop-mysql-1	mysql/mysql-server:8.0	"/entrypoint.sh mysql..."	
mysql	7 minutes ago	Up 7 minutes (healthy)	0.0.0.0:3306->3306/tcp, [::]:3306->3306/tcp
myshop-redis-1	redis:alpine	"docker-entrypoint.s..."	
redis	7 minutes ago	Up 7 minutes (healthy)	0.0.0.0:6379->6379/tcp, [::]:6379->6379/tcp

En la columna **PORTS**, busca la línea de `mysql`. Verás algo como `0.0.0.0:3306->3306/tcp`, donde el primer número (antes de `->`), en este caso `3306`, es el puerto expuesto en tu máquina local.

[!NOTE] Puerto por defecto Por defecto, Laravel Sail expone MySQL en el puerto **3306**. Sin embargo, si ya tienes otro servicio MySQL ejecutándose en tu sistema, Sail podría usar un puerto diferente (como 3307, 3308, etc.).

Una vez verificado el puerto, configura la conexión en DBCode:

### 2. Revisamos si se han realizado las migraciones:

### 3. Verificar que la Aplicación Funciona

Abre tu navegador web y visita <http://localhost>. Deberías ver la página de bienvenida de Laravel.

[!WARNING] Cuidado Si en lugar de ver la página de Laravel, ves la página de Apache de Ubuntu, es porque Apache está usando el mismo puerto.

- Parar Apache ahora `sudo systemctl stop apache2`
- Si quieres que no se inicie al arrancar el sistema: `sudo systemctl disable apache2`

[!IMPORTANT] Captura de pantalla requerida Realiza una captura de pantalla de la página inicial de Laravel funcionando en `http://localhost`. La captura debe incluir tu terminal con el prompt `usuario@equipo:~/ruta/proyecto$` visible.

4. Verificación del entorno

Una vez completada la configuración, es importante verificar que todos los servicios estén funcionando correctamente. Ejecuta los siguientes comandos para confirmar:

1. Verificar Estado de los Contenedores

```
sail ps
```

El comando `sail ps` muestra el estado actual de los contenedores Docker que utiliza tu entorno de desarrollo de Laravel Sail. Al ejecutarlo, verás una tabla con información como el nombre del contenedor, la imagen utilizada, el servicio que representa (por ejemplo, `laravel.test`, `mysql`, `redis`), el tiempo que llevan en ejecución y los puertos expuestos. Si todo está funcionando correctamente, los contenedores aparecerán con el estado "Up" o "running", lo que indica que los servicios (aplicación Laravel, base de datos MySQL, Redis, etc.) están activos y listos para usarse.

NAME	IMAGE	COMMAND	SERVICE
CREATED	STATUS	PORTS	
myshop-laravel.test-1	sail-8.4/app	"start-container"	
laravel.test	7 minutes ago	Up 7 minutes	0.0.0.0:80->80/tcp,
			[::]:80->80/tcp, 0.0.0.0:5173->5173/tcp, [::]:5173->5173/tcp
myshop-mysql-1	mysql/mysql-server:8.0	"/entrypoint.sh mysql..."	mysql
7 minutes ago	Up 7 minutes (healthy)	0.0.0.0:3306->3306/tcp, [::]:3306->3306/tcp	
myshop-redis-1	redis:alpine	"docker-entrypoint.s..."	redis
7 minutes ago	Up 7 minutes (healthy)	0.0.0.0:6379->6379/tcp, [::]:6379->6379/tcp	

2. Verificar Conexiones de Base de Datos

```
# Probar conexión MySQL
sail php artisan migrate:status
```

Al ejecutar el status nos mostrará que "tablas" estan creadas en MySQL.

```
Migration name ..... Batch /
Status
0001_01_01_000000_create_users_table .....
[1] Ran
```

```
0001_01_01_000001_create_cache_table .....  
[1] Ran  
0001_01_01_000002_create_jobs_table .....  
[1] Ran
```

## FASE 6: Frontend, Calidad de Código y Depuración

### 1. Frontend con Vite y Tailwind CSS

#### 1. Instalar dependencias de Node.js:

```
sail npm install
```

```
added 158 packages, and audited 159 packages in 12s  
  
30 packages are looking for funding  
run `npm fund` for details  
  
found 0 vulnerabilities
```

2. **Añadir estilos personalizados:** Abre el archivo `resources/css/app.css` y añade al final del archivo (después del contenido existente) los siguientes estilos personalizados:

```
/* Estilos personalizados para la tienda */  
.hero-gradient {  
  background: linear-gradient(135deg, #6997ca 0%, #4b7db2 100%);  
}  
  
.product-card {  
  transition: all 0.3s ease;  
}  
  
.product-card:hover {  
  transform: translateY(-5px);  
  box-shadow: 0 10px 25px rgba(0, 0, 0, 0.1);  
}  
  
.category-badge {  
  background: linear-gradient(135deg, #63d1c1 0%, #41b8a5 100%);  
}  
  
.stock-indicator {  
  position: relative;  
}  
  
.stock-indicator::before {
```

```

    content: '';
    position: absolute;
    left: -8px;
    top: 50%;
    transform: translateY(-50%);
    width: 6px;
    height: 6px;
    border-radius: 50%;
}

.stock-indicator.in-stock::before {
    background-color: #10b981;
}

.stock-indicator.low-stock::before {
    background-color: #ef4444;
}

.dark-mode-toggle {
    transition: all 0.3s ease;
}

.dark-mode-toggle:hover {
    transform: scale(1.1);
}

.mobile-menu {
    animation: slideDown 0.3s ease-out;
}

@keyframes slideDown {
    from {
        opacity: 0;
        transform: translateY(-10px);
    }
    to {
        opacity: 1;
        transform: translateY(0);
    }
}

```

[!IMPORTANT] No modifiques el contenido existente El archivo `app.css` ya contiene configuración importante de Tailwind CSS. Solo añade estos estilos al final del archivo, sin modificar las líneas que ya están.

3. **Abrir el archivo Blade principal:** Abre `resources/views/welcome.blade.php` y sustituye todo su contenido por el siguiente código:

```

<!DOCTYPE html>
<html lang="es">
<head>

```

```

<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Mi Tienda Online</title>
<script src="https://cdn.tailwindcss.com"></script>
<script>
  tailwind.config = {
    darkMode: 'class',
    theme: {
      extend: {
        colors: {
          primary: {
            500: '#6997ca',
            600: '#4b7db2',
            700: '#316298',
          },
          secondary: {
            500: '#63d1c1',
            600: '#41b8a5',
          },
        },
      },
    },
  },
}
</script>
@vite(['resources/css/app.css'])
</head>
<body class="bg-gray-50 dark:bg-gray-900">
  <!-- Header con navegación -->
  <header class="bg-white dark:bg-gray-800 shadow-lg relative">
    <div class="container mx-auto px-6 py-4">
      <div class="flex items-center justify-between">
        <!-- Logo -->
        <div class="flex items-center space-x-4">
          <a href="/" class="text-2xl font-bold text-primary-600
hover:text-primary-700 transition">🏠 Mi Tienda</a>
        </div>

        <!-- Navegación desktop -->
        <nav class="hidden lg:flex space-x-8">
          <a href="#" class="text-gray-700 dark:text-gray-300
hover:text-primary-600 transition">Inicio</a>
          <a href="#" class="text-gray-700 dark:text-gray-300
hover:text-primary-600 transition">Productos</a>
          <a href="#" class="text-gray-700 dark:text-gray-300
hover:text-primary-600 transition">Categorías</a>
          <a href="#" class="text-gray-700 dark:text-gray-300
hover:text-primary-600 transition">Ofertas</a>
          <a href="#" class="text-gray-700 dark:text-gray-300
hover:text-primary-600 transition">Contacto</a>
        </nav>

        <!-- Botones desktop -->
        <div class="hidden lg:flex items-center space-x-4">
          <button class="text-gray-700 dark:text-gray-300

```

```

hover:text-primary-600 transition">
     Carrito (0)
  </button>
  <button class="bg-primary-600 text-white px-4 py-2
rounded-lg hover:bg-primary-700 transition">
    Iniciar Sesión
  </button>
  <button class="border-2 border-primary-600 text-
primary-600 px-4 py-2 rounded-lg hover:bg-primary-600 hover:text-white
transition">
    Registrarse
  </button>
  <!-- Botón de modo oscuro desktop -->
  <button id="darkModeToggleDesktop" class="text-gray-700
dark:text-gray-300 hover:text-primary-600 transition p-2 rounded-full">
    
  </button>
</div>

<!-- Botones móvil/tablet -->
<div class="flex items-center space-x-2 lg:hidden">
  <!-- Botón de modo oscuro -->
  <button id="darkModeToggle" class="text-gray-700
dark:text-gray-300 hover:text-primary-600 transition p-2 rounded-full dark-
mode-toggle">
    
  </button>
  <!-- Botón menú móvil -->
  <button id="mobileMenuToggle" class="text-gray-700
dark:text-gray-300 hover:text-primary-600 transition">
    <svg class="w-6 h-6" fill="none"
stroke="currentColor" viewBox="0 0 24 24">
      <path stroke-linecap="round" stroke-
linejoin="round" stroke-width="2" d="M4 6h16M4 12h16M4 18h16"></path>
    </svg>
  </button>
</div>
</div>

<!-- Menú móvil -->
<div id="mobileMenu" class="lg:hidden hidden mt-4 pb-4
border-t border-gray-200 dark:border-gray-700 mobile-menu">
  <nav class="flex flex-col space-y-4 pt-4">
    <a href="#" class="text-gray-700 dark:text-gray-300
hover:text-primary-600 transition">Inicio</a>
    <a href="#" class="text-gray-700 dark:text-gray-300
hover:text-primary-600 transition">Productos</a>
    <a href="#" class="text-gray-700 dark:text-gray-300
hover:text-primary-600 transition">Categorías</a>
    <a href="#" class="text-gray-700 dark:text-gray-300
hover:text-primary-600 transition">Ofertas</a>
    <a href="#" class="text-gray-700 dark:text-gray-300
hover:text-primary-600 transition">Contacto</a>
    <div class="flex flex-col space-y-2 pt-4 border-t

```

```

border-gray-200 dark:border-gray-700">
    <button class="text-left text-gray-700 dark:text-
gray-300 hover:text-primary-600 transition">
        🛒 Carrito (0)
    </button>
    <button class="bg-primary-600 text-white px-4 py-2
rounded-lg hover:bg-primary-700 transition text-left">
        Iniciar Sesión
    </button>
    <button class="border-2 border-primary-600 text-
primary-600 px-4 py-2 rounded-lg hover:bg-primary-600 hover:text-white
transition text-left">
        Registrarse
    </button>
</div>
</nav>
</div>
</div>
</header>

<!-- Hero Section -->
<section class="hero-gradient text-white py-20">
    <div class="container mx-auto px-6 text-center">
        <h2 class="text-4xl md:text-6xl font-extrabold leading-tight mb-
6">
            Bienvenido a Mi Tienda
        </h2>
        <p class="text-xl md:text-2xl text-blue-100 mb-8 max-w-3xl mx-
auto">
            Descubre una amplia variedad de productos de calidad.
            Encuentra lo que buscas al mejor precio.
        </p>
        <div class="flex flex-wrap justify-center gap-4">
            <button class="bg-white text-primary-600 font-bold py-4 px-8
rounded-full hover:bg-gray-100 transition duration-300 ease-in-out transform
hover:scale-105">
                Ver Productos
            </button>
            <button class="border-2 border-white text-white font-bold
py-4 px-8 rounded-full hover:bg-white hover:text-primary-600 transition
duration-300 ease-in-out">
                Ofertas Especiales
            </button>
        </div>
    </div>
</section>

<!-- Categorías de Productos -->
<section class="py-16">
    <div class="container mx-auto px-6">
        <h3 class="text-3xl font-bold mb-12 text-center text-gray-900
dark:text-white">
            Nuestras Categorías
        </h3>

```

```

        <div class="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-4 gap-
8">

            <div class="bg-white dark:bg-gray-800 rounded-lg shadow-lg
p-6 product-card cursor-pointer">
                <div class="text-4xl text-primary-500 mb-4">📦</div>
                <h4 class="text-xl font-bold mb-2 text-gray-900
dark:text-white">Categoría 1</h4>
                <p class="text-gray-600 dark:text-gray-300 mb-4">
                    Descripción de la primera categoría de productos.
                </p>
                <button class="text-primary-600 font-semibold
hover:text-primary-700 transition">
                    Ver Productos →
                </button>
            </div>

            <div class="bg-white dark:bg-gray-800 rounded-lg shadow-lg
p-6 product-card cursor-pointer">
                <div class="text-4xl text-primary-500 mb-4">📦</div>
                <h4 class="text-xl font-bold mb-2 text-gray-900
dark:text-white">Categoría 2</h4>
                <p class="text-gray-600 dark:text-gray-300 mb-4">
                    Descripción de la segunda categoría de productos.
                </p>
                <button class="text-primary-600 font-semibold
hover:text-primary-700 transition">
                    Ver Productos →
                </button>
            </div>

            <div class="bg-white dark:bg-gray-800 rounded-lg shadow-lg
p-6 product-card cursor-pointer">
                <div class="text-4xl text-primary-500 mb-4">★</div>
                <h4 class="text-xl font-bold mb-2 text-gray-900
dark:text-white">Categoría 3</h4>
                <p class="text-gray-600 dark:text-gray-300 mb-4">
                    Descripción de la tercera categoría de productos.
                </p>
                <button class="text-primary-600 font-semibold
hover:text-primary-700 transition">
                    Ver Productos →
                </button>
            </div>

            <div class="bg-white dark:bg-gray-800 rounded-lg shadow-lg
p-6 product-card cursor-pointer">
                <div class="text-4xl text-primary-500 mb-4">🔗</div>
                <h4 class="text-xl font-bold mb-2 text-gray-900
dark:text-white">Categoría 4</h4>
                <p class="text-gray-600 dark:text-gray-300 mb-4">
                    Descripción de la cuarta categoría de productos.
                </p>
                <button class="text-primary-600 font-semibold

```

```

    hover:text-primary-700 transition">
      Ver Productos →
    </button>
  </div>

</div>
</div>
</section>

<!-- Productos Destacados -->
<section class="py-16 bg-gray-100 dark:bg-gray-800">
  <div class="container mx-auto px-6">
    <h3 class="text-3xl font-bold mb-12 text-center text-gray-900
dark:text-white">
      Productos Destacados
    </h3>
    <div class="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-
8">

      <div class="bg-white dark:bg-gray-700 rounded-lg shadow-lg
overflow-hidden product-card">
        <div class="h-48 bg-gray-200 dark:bg-gray-600 flex
items-center justify-center">
          <span class="text-4xl">📦 </span>
        </div>
        <div class="p-6">
          <h4 class="text-xl font-bold mb-2 text-gray-900
dark:text-white">Producto 1</h4>
          <p class="text-gray-600 dark:text-gray-300 mb-
4">Descripción del primer producto</p>
          <div class="flex items-center justify-between">
            <span class="text-2xl font-bold text-primary-
600">€XX</span>
            <button class="bg-primary-600 text-white px-4
py-2 rounded-lg hover:bg-primary-700 transition">
              Añadir al Carrito
            </button>
          </div>
        </div>
      </div>

      <div class="bg-white dark:bg-gray-700 rounded-lg shadow-lg
overflow-hidden product-card">
        <div class="h-48 bg-gray-200 dark:bg-gray-600 flex
items-center justify-center">
          <span class="text-4xl">📦 </span>
        </div>
        <div class="p-6">
          <h4 class="text-xl font-bold mb-2 text-gray-900
dark:text-white">Producto 2</h4>
          <p class="text-gray-600 dark:text-gray-300 mb-
4">Descripción del segundo producto</p>
          <div class="flex items-center justify-between">
            <span class="text-2xl font-bold text-primary-

```

```

600">€XX</span>
                <button class="bg-primary-600 text-white px-4
py-2 rounded-lg hover:bg-primary-700 transition">
                    Añadir al Carrito
                </button>
            </div>
        </div>
    </div>

    <div class="bg-white dark:bg-gray-700 rounded-lg shadow-lg
overflow-hidden product-card">
        <div class="h-48 bg-gray-200 dark:bg-gray-600 flex
items-center justify-center">
            <span class="text-4xl">★</span>
        </div>
        <div class="p-6">
            <h4 class="text-xl font-bold mb-2 text-gray-900
dark:text-white">Producto 3</h4>
            <p class="text-gray-600 dark:text-gray-300 mb-
4">Descripción del tercer producto</p>
            <div class="flex items-center justify-between">
                <span class="text-2xl font-bold text-primary-
600">€XX</span>
                <button class="bg-primary-600 text-white px-4
py-2 rounded-lg hover:bg-primary-700 transition">
                    Añadir al Carrito
                </button>
            </div>
        </div>
    </div>
</div>
</section>

<!-- Footer -->
<footer class="bg-gray-800 text-white py-12">
    <div class="container mx-auto px-6">
        <div class="grid grid-cols-1 md:grid-cols-4 gap-8">
            <div>
                <h5 class="text-xl font-bold mb-4">📦 Mi Tienda</h5>
                <p class="text-gray-400">
                    Tu tienda de confianza para encontrar los mejores
                    productos.
                </p>
            </div>
            <div>
                <h6 class="font-bold mb-4">Enlaces Rápidos</h6>
                <ul class="space-y-2 text-gray-400">
                    <li><a href="#" class="hover:text-white
transition">Sobre Nosotros</a></li>
                    <li><a href="#" class="hover:text-white
transition">Política de Privacidad</a></li>
                    <li><a href="#" class="hover:text-white

```

```

transition">Términos y Condiciones</a></li>
      <li><a href="#" class="hover:text-white
transition">Envíos y Devoluciones</a></li>
    </ul>
  </div>
  <div>
    <h6 class="font-bold mb-4">Atención al Cliente</h6>
    <ul class="space-y-2 text-gray-400">
      <li>☎ Teléfono de contacto</li>
      <li>✉ Email de contacto</li>
      <li>💬 Chat en vivo</li>
      <li>🕒 Horario de atención</li>
    </ul>
  </div>
  <div>
    <h6 class="font-bold mb-4">Síguenos</h6>
    <div class="flex space-x-4">
      <a href="#" class="text-gray-400 hover:text-white
transition">📘 Facebook</a>
      <a href="#" class="text-gray-400 hover:text-white
transition">📷 Instagram</a>
      <a href="#" class="text-gray-400 hover:text-white
transition">🐦 Twitter</a>
    </div>
  </div>
</div>
<div class="border-t border-gray-700 mt-8 pt-8 text-center text-
gray-400">
  <p>© 2025 Mi Tienda. Todos los derechos reservados.</p>
</div>
</div>
</footer>

<script>
  // Toggle dark mode functionality
  function toggleDarkMode() {
    document.documentElement.classList.toggle('dark');
    localStorage.setItem('darkMode',
document.documentElement.classList.contains('dark'));

    // Cambiar el icono según el modo para ambos botones
    const toggleButton = document.getElementById('darkModeToggle');
    const toggleButtonDesktop =
document.getElementById('darkModeToggleDesktop');

    if (document.documentElement.classList.contains('dark')) {
      if (toggleButton) toggleButton.innerHTML = '🌑';
      if (toggleButtonDesktop) toggleButtonDesktop.innerHTML =
'🌑';
    } else {
      if (toggleButton) toggleButton.innerHTML = '🌕';
      if (toggleButtonDesktop) toggleButtonDesktop.innerHTML =
'🌕';
    }
  }

```

```

    }

    // Toggle mobile menu functionality
    function toggleMobileMenu() {
        const mobileMenu = document.getElementById('mobileMenu');
        const menuToggle = document.getElementById('mobileMenuToggle');

        mobileMenu.classList.toggle('hidden');

        // Cambiar el icono del botón
        if (mobileMenu.classList.contains('hidden')) {
            menuToggle.innerHTML = `
                <svg class="w-6 h-6" fill="none" stroke="currentColor"
viewBox="0 0 24 24">
                    <path stroke-linecap="round" stroke-linejoin="round"
stroke-width="2" d="M4 6h16M4 12h16M4 18h16"></path>
                </svg>
            `;
        } else {
            menuToggle.innerHTML = `
                <svg class="w-6 h-6" fill="none" stroke="currentColor"
viewBox="0 0 24 24">
                    <path stroke-linecap="round" stroke-linejoin="round"
stroke-width="2" d="M6 18l18 6M6 6l12 12"></path>
                </svg>
            `;
        }
    }

    // Check for saved dark mode preference
    document.addEventListener('DOMContentLoaded', function() {
        if (localStorage.getItem('darkMode') === 'true') {
            document.documentElement.classList.add('dark');
            const toggleButton =
document.getElementById('darkModeToggle');
            const toggleButtonDesktop =
document.getElementById('darkModeToggleDesktop');
            if (toggleButton) toggleButton.innerHTML = '☀';
            if (toggleButtonDesktop) toggleButtonDesktop.innerHTML =
'☀';
        }

        // Configurar los botones
        const toggleButton = document.getElementById('darkModeToggle');
        const toggleButtonDesktop =
document.getElementById('darkModeToggleDesktop');
        if (toggleButton) toggleButton.onclick = toggleDarkMode;
        if (toggleButtonDesktop) toggleButtonDesktop.onclick =
toggleDarkMode;
        document.getElementById('mobileMenuToggle').onclick =
toggleMobileMenu;
    });
</script>

```

```
</body>  
</html>
```

#### 4. Compilar los assets:

Para que se cargue el CSS personalizado, ejecuta el siguiente comando para compilar los assets:

```
sail npm run build
```

[!TIP] Comandos de compilación para desarrollo continuo

Si se están haciendo cambios frecuentes en CSS/JS, se necesitaría mantener una consola abierta y ejecutar **sail npm run dev**. De este modo, se compilan los assets y se actualizan los cambios automáticamente.

Si vuelves a ir al **localhost**, deberías ver tu aplicación Laravel en funcionamiento con los estilos personalizados aplicados.

[!CAUTION] Captura de pantalla requerida **Personaliza tu Welcome**: Elige una temática para tu tienda online (por ejemplo: librería, tienda de ropa, electrónica, productos ecológicos, etc.).

Sustituye todos los datos estáticos del ejemplo (nombre de la tienda, secciones, productos, datos de contacto, redes sociales, etc.) por información relacionada con la temática que hayas elegido.

**No modifiques la estructura HTML ni el CSS**, solo cambia los textos, imágenes y datos para que reflejen tu tienda personalizada.

Ejemplo: Si eliges una tienda de libros, cambia "Mi Tienda" por el nombre de tu librería, los productos por libros, los datos de contacto por los tuyos, etc.

Así, tu página Welcome será única y adaptada a tu proyecto.

Realiza una captura de pantalla de la página o imprimela en PDF

## 2. Configurar herramienta

### Añadir Laravel Telescope

Laravel Telescope es una herramienta de depuración para Laravel que te permite monitorear solicitudes, excepciones, consultas de base de datos, eventos y más. Aquí te explico cómo integrarlo en tu proyecto:

1. Ejecuta este comando en la terminal dentro del directorio de tu proyecto:

```
sail composer require laravel/telescope --dev
```

```
INFO Discovering packages.
```

```
laravel/pail .....  
DONE  
laravel/sail .....  
DONE  
laravel/telescope .....  
DONE  
laravel/tinker .....  
DONE  
nesbot/carbon .....  
DONE  
nunomaduro/collision .....  
DONE  
nunomaduro/termwind .....  
DONE
```

```
81 packages you are using are looking for funding.
```

```
Use the `composer fund` command to find out more!
```

```
> @php artisan vendor:publish --tag=laravel-assets --ansi --force
```

```
INFO No publishable resources for tag [laravel-assets].
```

```
No security vulnerability advisories found.
```

```
Using version ^5.13 for laravel/telescope
```

```
sail php artisan telescope:install
```

```
Publishing Telescope Service Provider...
```

```
Publishing Telescope Configuration...
```

```
Publishing Telescope Migrations...
```

```
Telescope scaffolding installed successfully.
```

```
sail php artisan migrate
```

```
INFO Running migrations.
```

```
2025_09_30_140225_create_telescope_entries_table ..... 921.55ms
```

```
DONE
```

2. Una vez configurado, accede a Telescope en una nueva pestaña o ventana de tu navegador visitando [Telescope](#).
3. Al recargar la página inicial de Laravel verás cómo en la pantalla de Telescope te proporciona información sobre las rutas visitadas.

[!CAUTION] Captura de pantalla requerida Realiza una captura de pantalla de Telescope mostrando las rutas visitadas después de recargar la página inicial. La captura debe incluir tu terminal con el prompt `usuario@equipo:~/ruta/proyecto$` visible.

#### [!TIP] Consejo

Tómate tu tiempo para revisar las diferentes opciones que tiene Telescope. Es una herramienta muy útil y potente que puede que te ayude durante la realización de las futuras prácticas.

### Instalación de PHPStan para análisis de código estático:

1. Abre una terminal en la raíz de tu proyecto.
2. Ejecuta el siguiente comando para instalar PHPStan:

```
sail composer require --dev phpstan/phpstan
```

```
#...
82 packages you are using are looking for funding.
Use the `composer fund` command to find out more!
> @php artisan vendor:publish --tag=laravel-assets --ansi --force

INFO No publishable resources for tag [laravel-assets].

No security vulnerability advisories found.
Using version ^2.1 for phpstan/phpstan
```

3. Crear el archivo `phpstan.neon` en la raíz del proyecto:

```
nano phpstan.neon
```

4. Agrega la siguiente configuración:

```
parameters:
  level: 5
  paths:
    - app
  excludePaths:
    - tests/*
```

## 5. Ejecuta el análisis:

```
sail php vendor/bin/phpstan analyse
```

```
Note: Using configuration file /var/www/html/phpstan.neon.  
4/4 [████████████████████] 100%
```

```
[OK] No errors
```

[!CAUTION] Captura de pantalla requerida Realiza una captura de pantalla de tu terminal mostrando el comando `sail php vendor/bin/phpstan analyse` ejecutado correctamente sin errores. La captura debe incluir tu prompt `usuario@equipo:~/ruta/proyecto$` visible.

## Instalación de PHP\_CodeSniffer para validación de estilo:

### 1. Abre una terminal en la raíz de tu proyecto.

```
sail composer require --dev squizlabs/php_codesniffer
```

```
#...  
83 packages you are using are looking for funding.  
Use the `composer fund` command to find out more!  
> @php artisan vendor:publish --tag=laravel-assets --ansi --force  
  
INFO No publishable resources for tag [laravel-assets].  
  
No security vulnerability advisories found.  
Using version ^4.0 for squizlabs/php_codesniffer
```

### 2. Crea un archivo de configuración `.phpcs.xml` en la raíz:

```
nano .phpcs.xml
```

Añade la siguiente configuración:

```
<?xml version="1.0"?>  
<ruleset name="Laravel Coding Standard">  
  <rule ref="PSR12"/>  
</ruleset>
```

```
<file>app</file>
</ruleset>
```

### 3. Ejecuta la validación:

```
sail php vendor/bin/phpcs --standard=PSR12 app --report=summary
```

#### PHP CODE SNIFFER REPORT SUMMARY

FILE	ERRORS	WARNINGS
/var/www/html/app/Models/User.php	1	0

A TOTAL OF 1 ERROR AND 0 WARNINGS WERE FOUND IN 1 FILE

PHPCBF CAN FIX 1 OF THESE SNIFF VIOLATIONS AUTOMATICALLY

Time: 40ms; Memory: 12MB

[!CAUTION] Captura de pantalla requerida Realiza una captura de pantalla de tu terminal mostrando el comando `sail php vendor/bin/phpcs --standard=PSR12 app --report=summary` ejecutado. La captura debe incluir tu prompt `usuario@equipo:~/ruta/proyecto$` visible.

### 4. Para corregir automáticamente los errores de estilo, puedes usar el siguiente comando:

```
sail php vendor/bin/phpcbf --standard=PSR12 app
```

#### PHPCBF RESULT SUMMARY

FILE	FIXED	REMAINING
/var/www/html/app/Models/User.php	1	0

A TOTAL OF 1 ERROR WERE FIXED IN 1 FILE

Time: 50ms; Memory: 12MB

## Instalación de Laravel Pint para formatear:

1. Abre una terminal en la raíz de tu proyecto.

```
sail composer require --dev laravel/pint
```

```
#...
83 packages you are using are looking for funding.
Use the `composer fund` command to find out more!
> @php artisan vendor:publish --tag=laravel-assets --ansi --force

INFO No publishable resources for tag [laravel-assets].

No security vulnerability advisories found.
Using version ^1.25 for laravel/pint
```

2. Crea un archivo de configuración `pint.json` en la raíz del proyecto:

```
nano pint.json
```

```
{
  "preset": "laravel"
}
```

3. Si solo quieres verificar si el código necesita formato sin aplicar los cambios, usa la opción `--test`:

```
sail php vendor/bin/pint --test
```

```
.x.....
----- Laravel
FAIL ..... 29 files, 1 style
issue
x app/Models/User.php
class_attributes_separation
```

[!NOTE] Explicación del error Laravel Pint detecta como un problema de estilo cuando se usan varios `use` de traits dentro de una clase y no se deja una línea en blanco entre ellos. Es decir, cada declaración `use` de un trait debe estar separada por al menos una línea en blanco para cumplir con el estándar de estilo.

Por ejemplo, el siguiente código es incorrecto:

```
class User extends Authenticatable
{
    /** @use HasFactory<\Database\Factories\UserFactory> */
    use HasFactory;
    use Notifiable;

    // El resto del código
}
```

El código correcto sería:

```
class User extends Authenticatable
{
    /** @use HasFactory<\Database\Factories\UserFactory> */
    use HasFactory;

    use Notifiable;

    // El resto del código
}
```

La línea en blanco entre los **use** de traits es la que soluciona el error reportado por Pint.

Si después de ejecutar **sail up** te encuentras con errores de **"Permission Denied"** (Permiso denegado) al intentar ejecutar comandos o al cargar la web, es muy probable que los archivos de tu proyecto tengan permisos incorrectos.

Para solucionarlo, ejecuta estos comandos en la terminal de tu sistema (no dentro de Sail):

1. **Detener los contenedores:**

```
sail down
```

2. **Cambiar el propietario de los archivos** (sitúate en la carpeta de tu proyecto):

```
cd ~/development/laravel/MyShop
sudo chown -R $USER:$USER .
```

3. **Volver a levantar los contenedores:**

```
sail up -d
```

Esto asegura que tu usuario sea el propietario de todos los archivos, evitando conflictos con Docker. :::

```
.✓.....
----- Laravel
FIXED ..... 29 files, 1 style
issue fixed
✓ app/Models/User.php
```

[!CAUTION] Captura de pantalla requerida Realiza una captura de pantalla de tu terminal mostrando el comando `sail php vendor/bin/pint` ejecutado correctamente. La captura debe incluir tu prompt `usuario@equipo:~/ruta/proyecto$` visible.

## ENTREGA DE LA PRÁCTICA

### Requisitos de Entrega

Para completar esta práctica, debes entregar un **PDF** que incluya las siguientes capturas de pantalla:

#### 1. Laravel Sail - Contenedores Iniciados

- Captura de tu terminal mostrando el comando `sail up -d --build` ejecutándose correctamente
- Debe mostrar todos los contenedores iniciados (laravel.test, mysql, redis)
- La captura debe incluir tu prompt `usuario@equipo:~/ruta/proyecto$` visible

#### 2. Tienda Online Personalizada

- Captura de la página de bienvenida personalizada funcionando en `http://localhost`
- Debe mostrar la tienda online con la temática elegida (librería, ropa, electrónica, productos ecológicos, etc.)
- La página debe incluir:
  - Nombre de la tienda personalizado
  - Categorías de productos adaptadas a la temática
  - Productos específicos con nombres, descripciones y precios reales
  - Datos de contacto personalizados
  - Redes sociales actualizadas
- La página debe estar completamente funcional con Tailwind CSS aplicado

#### 3. Laravel Telescope - Rutas Visitadas

- Captura de Telescope mostrando las rutas visitadas después de recargar la página inicial
- Debe mostrar información de navegación a la tienda online personalizada
- La captura debe incluir tu terminal con el prompt `usuario@equipo:~/ruta/proyecto$` visible

#### 4. Análisis Estático con PHPStan

- Captura de tu terminal mostrando el comando `sail php vendor/bin/phpstan analyse` ejecutado correctamente
- Debe mostrar el análisis completado sin errores críticos
- La captura debe incluir tu prompt `usuario@equipo:~/ruta/proyecto$` visible

## 5. Validación de Estilo con PHP\_CodeSniffer

- Captura de tu terminal mostrando el comando `sail php vendor/bin/phpcs --standard=PSR12 app --report=summary` ejecutado
- Debe mostrar la validación de estilo completada
- La captura debe incluir tu prompt `usuario@equipo:~/ruta/proyecto$` visible

## 6. Formateo con Laravel Pint

- Captura de tu terminal mostrando el comando `sail php vendor/bin/pint` ejecutado correctamente
- Debe mostrar el formateo completado sin errores
- La captura debe incluir tu prompt `usuario@equipo:~/ruta/proyecto$` visible

## Instrucciones de Entrega

1. **Formato:** Entregar un único archivo PDF con todas las capturas
2. **Organización:** Incluir un título para cada captura explicando qué muestra
3. **Calidad:** Las capturas deben ser claras y legibles
4. **Compleitud:** Todas las capturas deben mostrar el funcionamiento correcto
5. **Personalización:** La tienda online debe reflejar claramente la temática elegida con contenido específico y realista

## Plataforma de Entrega

### Entrega en Aules