**Mostrar el listado de post**

**\*\*Antes de continuar se recomienda instalar tailwind css intellesense para VSC**

1. **Crear controlador para los Post**

**Una captura de pantalla de un celular con texto e imagen

Descripción generada automáticamente con confianza media**

1. **Luego codificamos dentro del controlador. Se deben llamar al modelo Post.**

**Texto

Descripción generada automáticamente**

1. **Retornamos dentro del método, la vista index**

**Texto

Descripción generada automáticamente**

1. **La ruta posts.index aún no existe. La creamos dentro de resources/views**

**Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente**

1. **En esta vista indicamos que queremos que se extienda nuestra plantilla principal**

**Imagen de la pantalla de un celular de un mensaje en letras blancas

Descripción generada automáticamente con confianza baja**

1. **Creamos la ruta respectiva**

**\*\*\* corrección: en el name de la ruta sería posts.index no post.index**

**Pantalla de computadora con letras

Descripción generada automáticamente con confianza media**

1. **Actualizamos la página y solo veremos la plantilla.**
2. **Ahora implementamos un div containter dentro de este llamado del componente. Para ello, creamos un div de color (el color que se desee) así:**

**Pantalla negra con letras blancas

Descripción generada automáticamente**

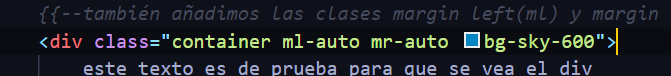
**Probamos en el navegador y veremos un div de color.**

**\*\*\*\*\*\*Si las clases de tailwind no funcionan, es necesario correr nuevamente npm run dev. En algunos casos, adicional a ello, es necesario borrar las cookies del navegador.**

**Texto

Descripción generada automáticamente**

1. **Si quisiéramos simplemente centrar sin pensar en el ancho específico del div, digitamos lo siguiente:**

****

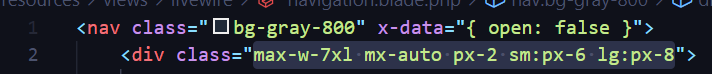
**Pero si recargamos el navegador web, veremos que no luce centrado adecuadamente en relación al menú de la parte superior. Para solucionar esto, hacemos lo siguiente:**

1. **Vamos a navigation**

**Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media**

**Copiamos los atributos de la segunda línea**

****

1. **Y los pegamos dentro de nuestro div container en el index**

**Una captura de pantalla de un celular

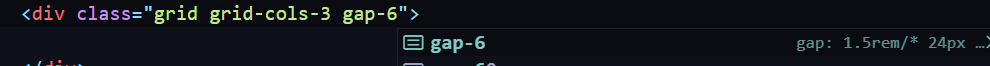
Descripción generada automáticamente con confianza media**

1. **Ahora dentro del div vamos a crear un grid (El grid es una cuadrícula sobre la que se distribuyen los distintos elementos que componen la web). Podemos escribir .grid para obtenerlo de manera automática con su respectiva clase.**

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

Descripción generada automáticamente**

1. **Indicamos el número de columnas con col y además la separación (gap) que queremos que haya entre cada columna. Con gap-6 estaríamos indicando una separación de 1.5 rem**

****

1. **Creamos un foreach para poder iterar los elementos del array $posts del controlador y allí mostraremos las imágenes. Para poder mostrar dichas imágenes que están en la tabla images, es necesario recurrir a la relación anteriormente creada que está en el modelo Post**

**Texto

Descripción generada automáticamente**

**Texto

Descripción generada automáticamente**

**\*\*Sin embargo, aquí hay un error, porque la clase/modelo images no existe. Realmente, el modelo invocado se llama image. De hecho al ser una clase, debió llamarse Image.**

**La relación necesaria se llama image**

**Interfaz de usuario gráfica

Descripción generada automáticamente**

**Texto

Descripción generada automáticamente**

**Al recargar la página web, observaremos una gran cantidad de información de los posts. Pero solamente necesitamos la url de las imágenes.**

**\*\*\*\*en caso de que no se visualicen los datos en 3 columnas, será necesario correr una vez más el comando npm run dev.**

1. **Para cargar solo las imágenes de los posts lo indicamos así:**

**Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente**

**Pantalla de computadora con letras

Descripción generada automáticamente con confianza media**

1. **Ahora si colocamos una ruta específica de estas que estamos viendo en pantalla, al lado de la ruta principal del proyecto, no se visualizará ninguna imagen:**

**Texto

Descripción generada automáticamente**

**Pero si le agregáramos la ruta storage, sí visualizaríamos la imagen:**

**Interfaz de usuario gráfica, Sitio web

Descripción generada automáticamente**

1. **Lo que debemos hacer como en otros proyectos es usar el método Storage::url(), y dentro del paréntesis colocar lo que habíamos digitado:**

**Interfaz de usuario gráfica, Texto

Descripción generada automáticamente**

**En ese caso sí veremos la url completa de la imagen:**

**Captura de pantalla de computadora

Descripción generada automáticamente**

1. **Luego, para ver las imágenes, las incluimos en un img de la siguiente manera:**

**Texto

Descripción generada automáticamente**

1. **No obstante, pudiera ser que si no tenemos personalizado un dominio para el proyecto, las imágenes no se muestren. En ese caso, es necesario ir al archivo .env y configurar la APP\_URL exactamente como la estamos usando actualmente, con :8000, así:**

**Interfaz de usuario gráfica

Descripción generada automáticamente con confianza baja**

**Captura de pantalla de un celular

Descripción generada automáticamente**

1. **Quitamos el color de fondo**

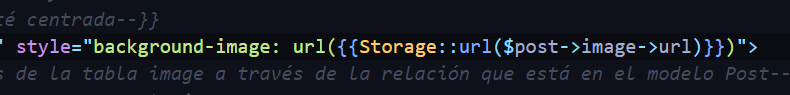
**Para escribir encima de la imagen, debemos hacer un cambio**

**Debido a que necesitamos escribir encima de la imagen, vamos a cambiar el método para mostrar las imágenes. En este caso sería usando un estilo css incrustado en el article, así:**

**Texto

Descripción generada automáticamente**

**La captura anterior está recortada. A continuación, el faltante de la derecha:**

****

**\*\*\*en caso de que las imágenes no se vean, será necesario correr una vez más npm run dev (investigar más a fondo sobre este comando).**