PROYECTO PÁGINA PERSONAL

Si eres desarrollador, debes tener una página web personal, eso es ley.

Este proyecto es a base de HTML con Tailwind CSS y ha sido el mejor comienzo para soltar la mano.

¿Qué es Tailwind CSS?

Tailwind CSS, son un montón de clases ya fabricadas para llegar e invocar, a tu gusto.

Partamos del comienzo.

HTML hoy en día es muy simple, te lo muestro a continuación.

Hacemos un contenedor con etiqueta <div> y si le escribimos adentro “Hola mundo”.

<!DOCTYPE html>

<html lang="es">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <!-- esta parte es la importante -->

    <div>

        Hola mundo

    </div>

    <!-- y hasta aquí -->

</body>

</html>

Automáticamente ese hola mundo se renderiza como un párrafo, es decir, etiqueta <p>, y te lo escribe con fuente predeterminada de sistema.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Pero si pensamos a largo plazo una etiqueta <p> no tiene tanto peso para el SEO (Search Engine Optimization) de Google Analytics, y hay jerarquías para organizar el contenido así que cambiaremos la etiqueta <p> implícita, por un título <h1> para aumentar el ranking en los resultados de búsquedas, aumentar la suerte para que nuestra página aparezca enm la primera página de Google, y le pondremos una clase de estilos CSS llamada “primeraclase” que después escribiremos todo el estilo que tendrá esa clase.

HTML quedaría así:

    <!-- esta parte es la importante -->

    <div class="primeraclase">

        <h1>Hola mundo</h1>

    </div>

    <!-- y hasta aquí -->

Y el archivo CSS lo insertaremos en la parte <head> del HTML (más arriba), encapsulando este código con la etiqueta <style>, para así no crear un archivo aparte (por ejemplo, style.css al lado de ejemplo2.html), y quedaría así:

<!DOCTYPE html>

<html lang="es">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

    <style>

        /\* Estos son keyframes que la clase .primeraclase va a ocupar \*/

        @keyframes mover {

            0%,

            100% {

                transform: translate(0, 0);

            }

            25% {

                transform: translate(100%, -100%);

            }

            50% {

                transform: translate(-100%, 100%);

            }

            75% {

                transform: translate(100%, 100%);

            }

        }

        @keyframes cambiarColor {

            0% {

                color: #ffffff;

            }

            20% {

                color: #2b2525;

            }

            40% {

                color: #00c3ff;

            }

            60% {

                color: #FFFF00;

            }

            80% {

                color: #FF00FF;

            }

            100% {

                color: #00FFFF;

            }

        }

        @keyframes moverYgirar {

            0% {

                transform: translate(0, 0) rotate(0deg);

            }

            25% {

                transform: translate(20px, -30px) rotate(20deg);

            }

            50% {

                transform: translate(-20px, 20px) rotate(-20deg);

            }

            75% {

                transform: translate(30px, 0) rotate(20deg);

            }

            100% {

                transform: translate(0, 0) rotate(-20deg);

            }

        }

        /\* Aquí está la clase propiamente tal \*/

        /\* Configuramos tamaño, ancho y alto \*/

        .primeraclase {

            position: relative;

            width: 100%;

            height: 200px;

            background-color: #000;

            overflow: hidden;

            display: flex;

            align-items: center;

            justify-content: center;

        }

        /\* Este es un ejemplo de un selector descendente en CSS\*/

        /\* Aplica estilos al elemento h1 que sea descendiente de algo con la clase .primeraclase\*/

        .primeraclase h1 {

            position: relative;

            z-index: 2;

            /\* Asegura que el texto esté por encima de las pelotas \*/

            animation: cambiarColor 5s infinite, moverYgirar 2s infinite;

            margin: 10px;

        }

        /\* Aqui al before y el after se ocupan para crear pseudo-elementos y no anadir más HTML \*/

        .primeraclase::before,

        .primeraclase::after {

            content: '';

            position: absolute;

            width: 20px;

            height: 20px;

            border-radius: 50%;

            animation: mover 8s infinite alternate ease-in-out;

        }

        .primeraclase::before {

            background-color: #FF0000;

            /\* Color para la primera pelota \*/

        }

        .primeraclase::after {

            background-color: #00FF00;

            /\* Color para la segunda pelota \*/

            animation-delay: 4s;

            /\* Desfase para que no se muevan sincronizadamente \*/

        }

    </style>

</head>

<body>

    <!-- esta parte es la importante -->

    <div class="primeraclase">

        <h1>Hola mundo</h1>

    </div>

    <!-- y hasta aquí -->

</body>

</html>

Sí, es un tremendo código CSS, de CSS puro.

Y nos quedaría renderizado así:



Y aquí viene la pregunta importante

**Habitualmente ¿Necesitas un CSS tan personalizado?**

La respuesta es obviamente no, y por eso es tan genial **Tailwind CSS.**

Si habitualmente ocupas clases como “tamanofuente”, “contenedorfull”, “hoveramarillo”, es decir, tamaño de fuentes, ancho de contenedores, que tenga fondo negro, que tenga un hover al pasar el mouse encima del elemento, y sobretodo, lo más exquisto, hacer responsive la web rápidamente según el ancho del dispositivo (es decir que se muestre de una forma en tu celular mobile y de otra forma en desktop), Tailwind CSS te da una mano rápida y elegante.

Invocas el CDN de Tailwind dentro del <head> de nuestro HTML con:

<link href="https://cdn.jsdelivr.net/npm/tailwindcss@2.2.19/dist/tailwind.min.css" rel="stylesheet">

Y el resto de nuestro HTML quedaría:

<!DOCTYPE html>

<html lang="es">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

    <link href="https://cdn.jsdelivr.net/npm/tailwindcss@2.2.19/dist/tailwind.min.css" rel="stylesheet">

</head>

<!-- Aqui le aplica estilos a todo el body, es decir a todo el fondo de la página y el centrado de los elementos -->

<body class="bg-gradient-to-r from-green-300 via-blue-500 to-purple-600 h-screen flex items-center justify-center">

    <!-- Aquí por ejemplo w-full significa width:100% y ocupa todo el ancho de pantalla -->

    <div class="primeraclase w-full px-4 sm:max-w-xl md:max-w-2xl lg:max-w-4xl mx-auto py-8">

        <!-- Aquí notar el uso de breakpoints segun el ancho de pantalla, sm para pantallas pequenas, md medianas, lg grandes, y va aumentando o disminuyendo el texto -->

        <h1 class="text-white text-2xl sm:text-4xl md:text-5xl lg:text-6xl font-bold text-center text-2xl sm:text-3xl md:text-6xl lg:text-9xl shadow-indigo-500/50">Hola mundo</h1>

    </div>

</body>

</html>

Tailwind es otro tipo de CSS, para nada, es un CSS de toda la vida pero ya cocinado, es decir **text-2xl sm:text-3xl md:text-6xl lg:text-8xl** significa que se activa el breakpoint **sm:text-4xl** en caso que la pantalla de al menos 640px de ancho y ahí le aplica un tamaño de fuente de 2.25rem, si fuese de 639px de ancho se activa la clase antes del breakpoint, es decir, **text-2xl,** que son 1.5rem, y a pesar que la medida rem sea relativa, en la práctica no sirve para garantizar que se vea bien en cualquier pantalla, por lo que con el breakpoint **md:text-6x**l se activa para pantallas mayores con un mínimo de ancho de 768px, típico breakpoint que separa mobile de desktop, y el text-6xl significa aplicarle tamaño de fuente 3.75rem. Lo mismo para pantallas más grandes, con **lg:text-9xl,** lg: es breakpoint para pantallas de 1024px o más, y text-9xl es clase que aplica tamaño de fuente de 6 rem.

Para terminar de aclararlo comparemos CSS Puro con el CSS cocinado de Tailwind CSS:

**Comparación CSS Puro Vs CSS Tailwind**

<h1 class="sm:text-4xl md:text-6xl lg:text-8xl">Hola mundo</h1>

Esto vimos en el ejemplo, y sería una pura línea con CSS Tailwind.

Si lo replicamos con CSS puro y clásico:

Archivo HTML:

<h1 class="titulo-responsive">Hola mundo</h1>

Y el CSS con la clase titulo-responsive:

.titulo-resposive {

    font-size: 1.25rem; /\* Valor por defecto si no hay un tamaño específico antes de sm: \*/

}

@media (min-width: 640px) {

    .titulo-resposive {

        font-size: 2.25rem; /\* Equivalente a text-4xl en Tailwind \*/

    }

}

@media (min-width: 768px) {

    .titulo-resposive {

        font-size: 4rem; /\* Equivalente a text-6xl en Tailwind \*/

    }

}

@media (min-width: 1024px) {

    .titulo-resposive {

        font-size: 6rem; /\* Equivalente a text-8xl en Tailwind \*/

    }

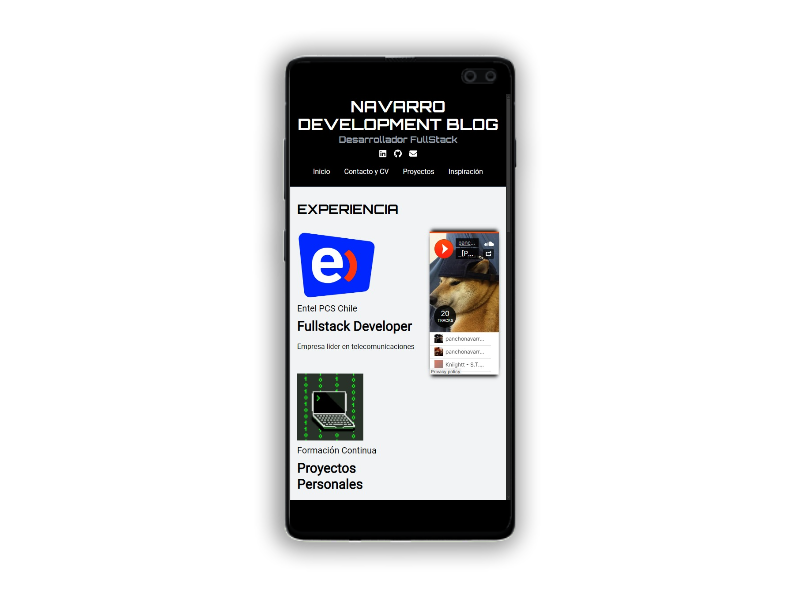
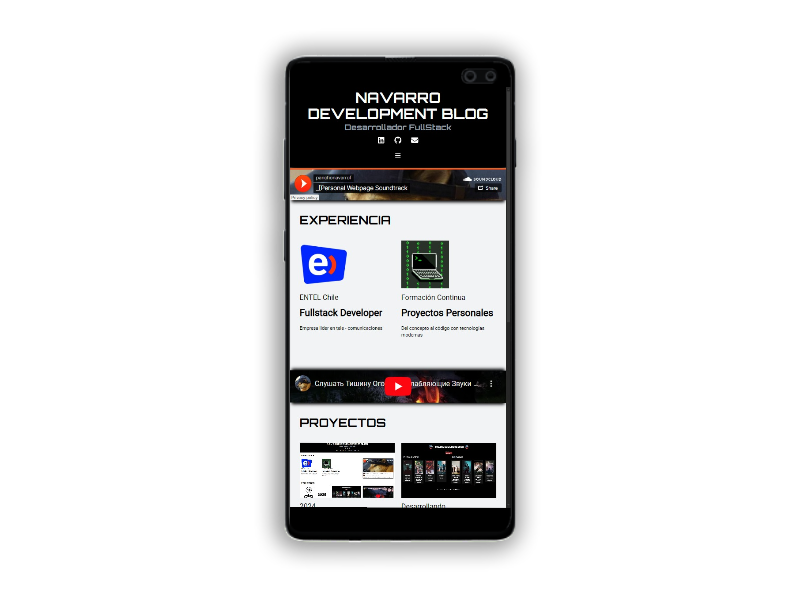
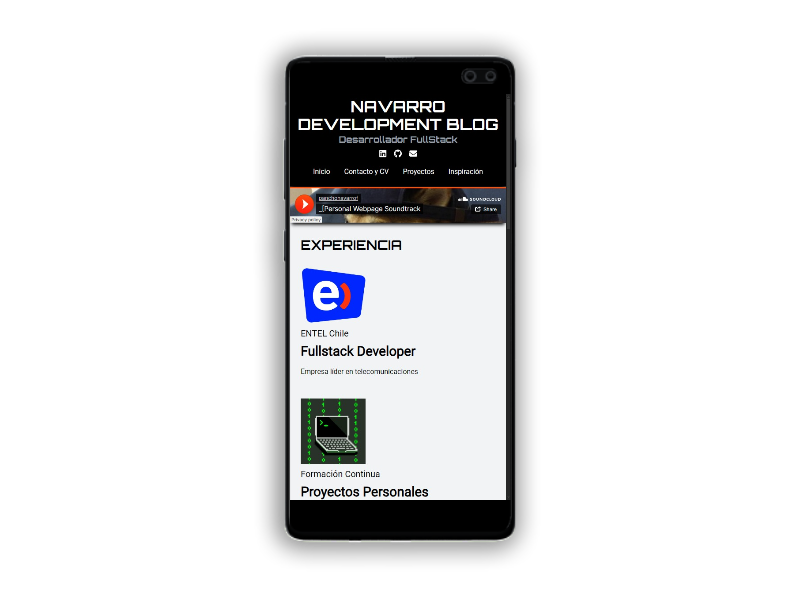
}

No es más difícil, pero aumenta el tiempo de desarrollo.

Y queremos avanzar rápido siempre.

Conclusión: Vale la pena familiarizarse con Tailwind CSS.

**Evolución de Versiones Móviles:**

****

Resumen:

* Primera versión era una versión comprimida de la escritorio. Link:
* Segunda versión el reproductor Soundcloud y el video Youtube son barras horizontales arriba de los títulos. La música no se puede ir. Link:
* Tercera versión se comprimen los elementos dentro de las secciones para que sean una galería de 2x2, y se agrega un separador a la sección de Blog, con el mismo estilo de los reproductores pero con una imagen gif.

**Ideas desde otros repositorios**

Como creo que la actual página de post está muy simple, necesito ideas, y aquí la IA no satisface las necesidades, necesitas hacer una búsqueda de repositorios en github.

Hacemos una búsqueda “post template” y filtramos por lenguaje HTML:

<https://github.com/search?q=post+template+language%3AHTML&type=repositories&l=HTML>

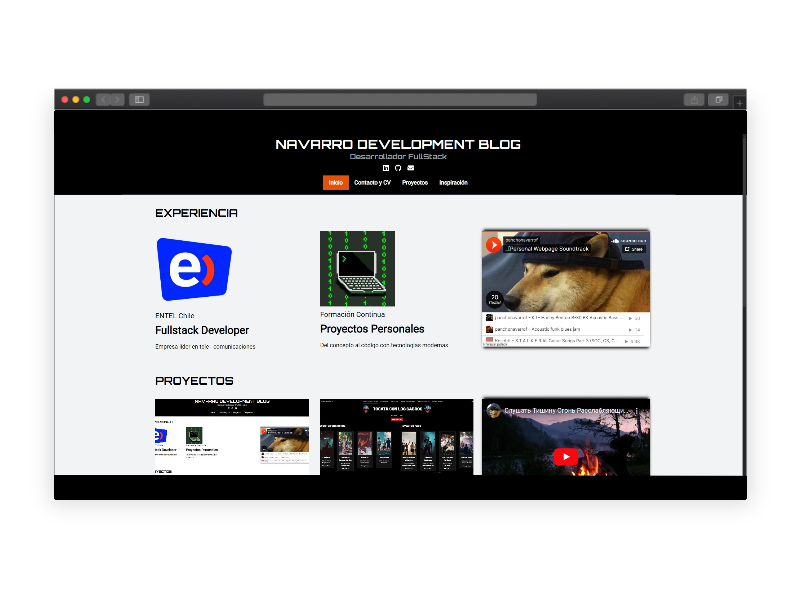
Encontramos los siguientes buenos:

<https://github.com/davidgrzyb/tailwind-blog-template> // de este repo me gusta la idea de un paginador al final de la entrada con las entradas siguientes como anteriores y también la idea de una card donde está la imagen del autor con sus enlace de redes sociales.

<https://github.com/Mehmet-github06/Informative-Web-Page> // de este repo sacamos la idea de señalar la página activa con un color de fondo en el navbar, es decir, página principal se activa la pestaña Inicio, con una señal visual, y si estás en la página Proyectos, esté marcado de color naranjo la pestaña Proyectos

<https://github.com/AndrewKoliaka/medium_blog> // y de este repo copiaremos, o nos inspiraremos, el cómo será propiamente tal la disposición de imágenes y textos en todas las páginas post.

**Evolución de Versiones Desktop:**

Estamos con esta versión 

Siento que es muy grande, se garantiza la accesibilidad, pero podría agruparse como una página de logros de un menú de videojuegos. Y lo podría conseguir con como quedó la versión mobile pero dispuesto con tres columnas.

Y algo muy importante, ¡falta el About Me!, no soy un robot, soy una persona y debo venderme como tal, consumo muchas series y películas, y sí, también veo anime, pero me defiendo diciendo que amo mucho más a los videojuegos.