

# Introducción

---

Vamos a crear un mini proyecto, que nos permita:

- Mostrar un listado de películas.
- Editar una película.
- Crear una película.
- Eliminar una película.

Todo con sus correspondientes *te toca* con la parte de actores para que practiques.

No nos vamos a ayudar de frameworks, ni de herramientas para poder gestionar mejor los estilos, verás que cierto código te podrá sonar repetitivo o difícil de mantener, ¿Por qué este "sufrimiento"? Para que cuando uses un Framework o ayudas, sepas que es lo que te aporta.

## Creación estructura básica del proyecto

---

- Partimos del sandbox de TypeScript.
- Antes de nada, vamos a modificar un poco nuestro proyecto.
  - Borramos los ficheros que no vamos a utilizar, *dummy.spec.ts* y *main.ts* y *style.css*.
  - Vamos a mover el fichero *index.html* a la carpeta *src*.
  - Vamos a añadir la siguiente configuración a nuestro *vite.config.ts*, para indicarle que *index.html* está ahora dentro de la carpeta *src* y va a ser nuestra página principal.

*./vite.config.ts*

```
export default defineConfig({
  plugins: [checker({ typescript: true })],
  test: vitestConfig.test,
  + root: "./src"
});
```

## Estilado

---

- Lo primero que vamos a hacer es crearnos una hoja de estilos de tipo "reset", ¿Qué quiere decir esto?
  - Cada navegador tiene sus propios estilos por defectos (márgenes, fuentes...).
  - Con esta hoja lo que hacemos es que tengamos el mismo punto de partida para cada navegador.

Existen proyectos open source que ya te hacen esto.

*./src/reseteo.css*

```
* {  
  box-sizing: border-box;  
}
```

```
html,  
body,  
div,  
span,  
applet,  
object,  
iframe,  
h1,  
h2,  
h3,  
h4,  
h5,  
h6,  
p,  
blockquote,  
pre,  
a,  
abbr,  
acronym,  
address,  
big,  
cite,  
code,  
del,  
dfn,  
em,  
img,  
ins,  
kbd,  
q,  
s,  
samp,  
small,  
strike,  
strong,  
sub,  
sup,  
tt,  
var,  
b,  
u,  
i,  
center,  
dl,  
dt,  
dd,  
ol,  
ul,  
li,
```

```
fieldset,
form,
label,
legend,
table,
caption,
tbody,
tfoot,
thead,
tr,
th,
td,
article,
aside,
canvas,
details,
embed,
figure,
figcaption,
footer,
header,
hgroup,
menu,
nav,
output,
ruby,
section,
summary,
time,
mark,
audio,
video {
    margin: 0;
    padding: 0;
    border: 0;
    font-size: 100%;
    font: inherit;
    vertical-align: baseline;
}
/* HTML5 display-role reset for older browsers */
article,
aside,
details,
figcaption,
figure,
footer,
header,
hgroup,
menu,
nav,
section {
    display: block;
}
body {
```

```

    line-height: 1;
  }
  ol,
  ul {
    list-style: none;
  }
  blockquote,
  q {
    quotes: none;
  }
  blockquote:before,
  blockquote:after,
  q:before,
  q:after {
    content: "";
    content: none;
  }
  table {
    border-collapse: collapse;
    border-spacing: 0;
  }

```

Ahora vamos a definir una hoja de estilo global para nuestra aplicación, en esta hoja meteremos todos los estilos de la aplicación.

Como verás esto del CSS irá creciendo, te va a resultar complicado de gestionar, y aunque dividiéramos en varios ficheros, es muy fácil que nombres de clases se repitan y nos den problemas, en proyectos reales existen soluciones que nos permiten encapsular estilos, como por ejemplo CSS Modules.

Vamos a arrancar por definir algunos estilos básicos.

Lo que definimos:

- Nos importamos una fuente de la CDN de google, en concreto la fuente *Roboto*.
- Definimos una serie de custom properties, es decir variables HTML, aquí guardamos valores de por ejemplo colores que vamos a usar en varios sitios y no queremos tener que repetir el mismo valor una y otra vez en nuestro CSS.
- Estilamos nuestro body, headings, links y botones.
- Creamos una clase para el contenedor principal de nuestra aplicación, que será el que tenga el tamaño de la pantalla y que nos permita centrar el contenido.

`./src/estilos.css`

```

@import url("https://fonts.googleapis.com/css2?
family=Roboto:wght@400;500;700&display=swap");

:root {
  --bg-color: rgb(229, 234, 238);
  --color-primary: #00ad74;

```

```
--color-secondary: #008c86;
--font-family: Roboto, "Arial", "Helvetica", sans-serif;
}

body {
  height: 100vh;
  font-family: var(--font-family);
  line-height: 1;
  background-color: var(--bg-color);
  overflow: hidden;
}

h1 {
  font-size: 2.5rem;
  font-weight: 700;
}

h2 {
  font-size: 2rem;
  font-weight: 600;
}

h3 {
  font-size: 1.5rem;
  font-weight: 500;
}

a {
  text-decoration: none;
  color: inherit;
  cursor: pointer;
}

button {
  display: inline-block;
  padding: 12px 24px;
  background-color: var(--color-primary);
  color: #fff;
  cursor: pointer;
  border: none;
  border-radius: 4px;
  font-size: 16px;
  font-weight: bold;
  text-align: center;
  text-decoration: none;
  transition: background-color 0.3s ease;
}

button:hover {
  background-color: var(--color-secondary);
}

.root {
  display: flex;
```

```

    flex-direction: column;
    align-items: center;
    justify-content: center;
    gap: 20px;
    height: 100vh;
}

.main {
    display: flex;
    flex-direction: column;
    gap: 30px;
    height: 100vh;
    overflow-y: scroll;
    position: relative;
}

```

En el HTML principal:

- Importamos los ficheros de hojas de estilo que acabamos de crear.
- Añadimos dos enlaces, uno para navegar a el listado de películas y otro para navegar el listado de actores:

```

<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <link rel="icon" type="image/svg+xml" href="/vite.svg" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
+   <link rel="stylesheet" href="reseteo.css">
+   <link rel="stylesheet" href="estilos.css">
-   <title>Vite + TS</title>
+   <title>Películas</title>
  </head>
  <body>
+   <div class="root">
+     <h1>Películas</h1>
+     <div>
+       <button>
+         <a href="./pelicula-listado/index.html">Listado Películas</a>
+       </button>
+       <button>
+         <a href="#">Listado Actores</a>
+       </button>
+     </div>
+   </div>
-   <script type="module" src="/src/main.ts"></script>
  </body>
</html>

```

- Vamos a ejecutar la aplicación:

```
npm run dev
```

# Películas

[Listado Películas](#)[Listado Actores](#)

Si pinchamos en los enlaces verás que nos da un error (intenta navegar a una página que no existe).



## No se puede encontrar esta página (127.0.0.1)

No se ha encontrado ninguna página web para la dirección  
**<http://127.0.0.1:5173/pelicula-listado/pelicula-listado.html>**.

HTTP ERROR 404

[Volver a cargar](#)

Hemos generado la siguiente estructura

```
src
├── index.html // Página principal (te da a elegir entre peliculas y actores)
├── estilos.css
└── reset.css
```

En el siguiente paso vamos a crear la página de listado.