

Práctica guiada Portfolio [Opcional]

Tras nuestro primer reto con Vite vamos a crear nuestra primera SPA a través de un Portfolio que podremos utilizar para mostrar nuestros proyectos a partir de ahora. Este Portfolio está basado en un diseño de Figma, el cual en un futuro podéis modificar (os recomendamos que personalicéis al máximo y lo hagáis vuestro).

Portfolio Ui Design Template (Community)

Created with Figma

[https://www.figma.com/file/ppmdnfmkZKavE98onmopKe/Portfolio-Ui-Design-Template-\(Community\)?node-id=61%3A2525](https://www.figma.com/file/ppmdnfmkZKavE98onmopKe/Portfolio-Ui-Design-Template-(Community)?node-id=61%3A2525)



Peter Parker

Hey, I'm

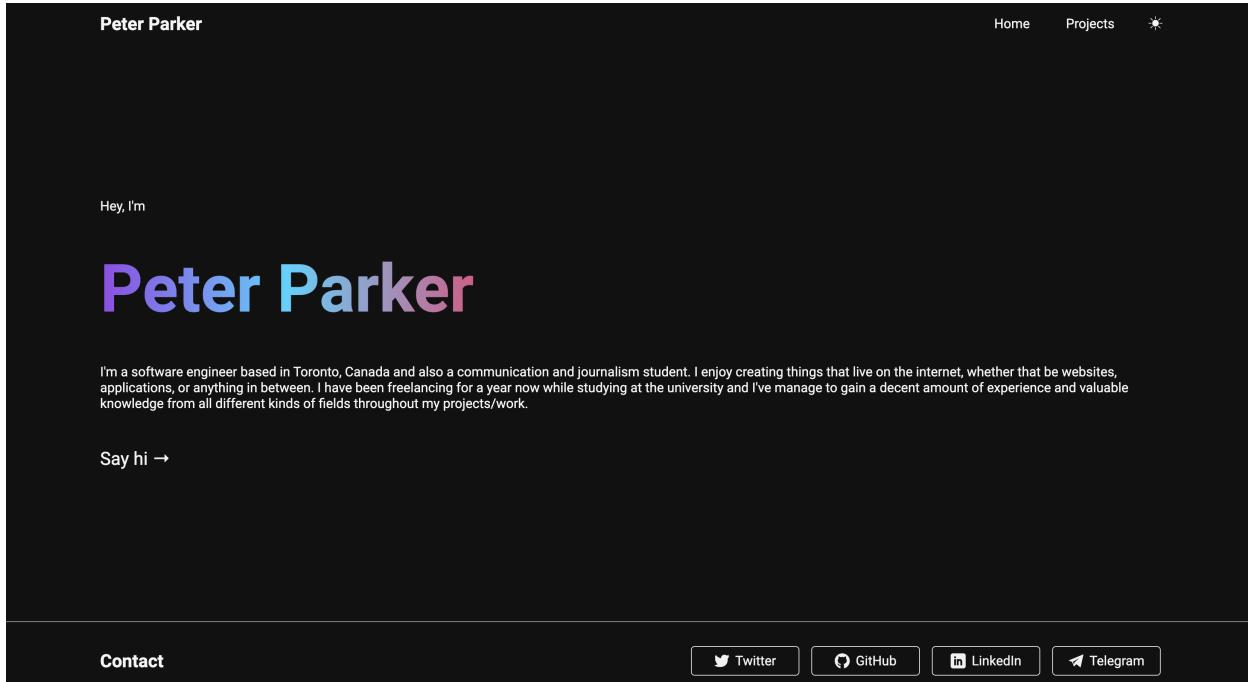
Peter Parker

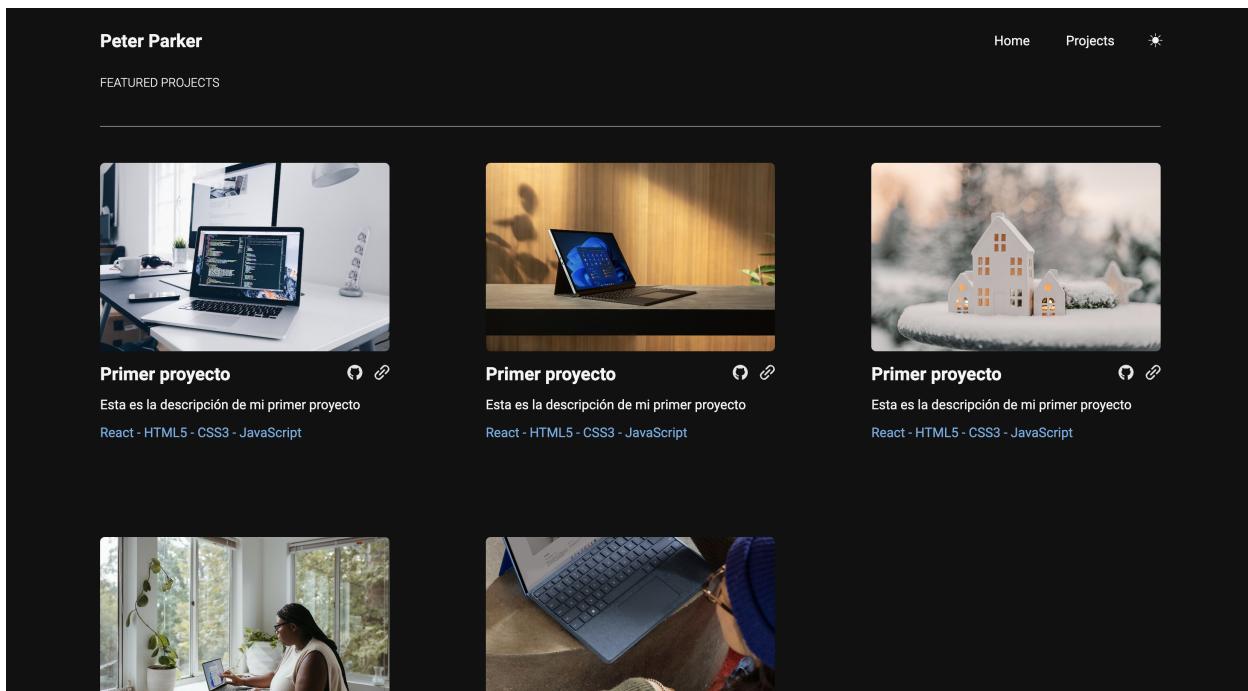
I'm a software engineer based in Toronto, Canada and also a communication and journalism student. I enjoy creating things that live on the internet, whether that be websites, applications, or anything in between. I have been freelancing for a year now while studying at the university and I've managed to gain a decent amount of experience and valuable knowledge from all different kinds of fields throughout my projects/work.

Say hi →

Contact

[Twitter](#) [GitHub](#) [LinkedIn](#) [Telegram](#)





Setup

Para comenzar con nuestra configuración, vamos a crear una nueva aplicación con **Vite**.

```
npm create vite@latest
✓ Project name: my-portfolio
✓ Select a framework: > vanilla
✓ Select a variant: > javascript
```

E instalaremos las dependencias necesarias:

```
cd my-portfolio
npm install
```

En este punto limpiaremos nuestro proyecto:

- Eliminaremos el fichero **javascript.svg** que contiene el logo de JS.
- Limpiaremos el fichero **style.css** dejándolo vacío para insertar nuestros propios estilos. Solo vamos a respetar los estilos anidados en **:root** para tener nuestras variables con las fuentes, los tamaños por defecto y el fondo.
- Eliminaremos el fichero **counter.js**.
- En el fichero **main.js** solo dejaremos la línea de importación de **style.css**

Vamos a modificar nuestro fichero **index.html** para modificar el título y colocar las etiquetas "header" y "footer" para cumplimentarla de manera dinámica a través de nuestro script:

```

<!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" />
    <title>My Portfolio</title>
  </head>
  <body>
    <header></header>
    <main></main>
    <footer></footer>
    <script type="module" src="/main.js"></script>
  </body>
</html>

```

Estilos globales y assets

Antes de empezar, vamos a definir unos estilos globales en el fichero "style.css" basados en una paleta de colores oscura y clara, la cual podremos intercambiar mediante una clase llamada ".light".

Le implementaremos además una fuente externa y un pequeño reset de estilos por defecto a eliminar:

```

@import url("https://fonts.googleapis.com/css2?family=Roboto:wght@300;400;500;700;900&display=swap");

:root {
  --primary-dark: whitesmoke;
  --bg-dark: #111111;
  --primary-light: #111111;
  --bg-light: whitesmoke;
  --highlight: #6cace4;
  --divider: rgb(164, 164, 164);
}

* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
  font-family: "Roboto", sans-serif;
}

li {
  list-style-type: none;
}

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

html {
  scroll-behavior: smooth;
}

body {
  background-color: var(--bg-dark);
  color: var(--primary-dark);
  transition: all 0.5s ease-in-out;
}

.light {
  background-color: var(--bg-light);
  color: var(--primary-light);
}

main {
  min-height: 80vh;
}

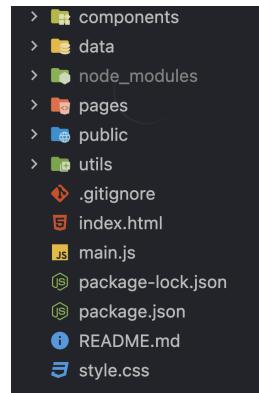
a,
p {

```

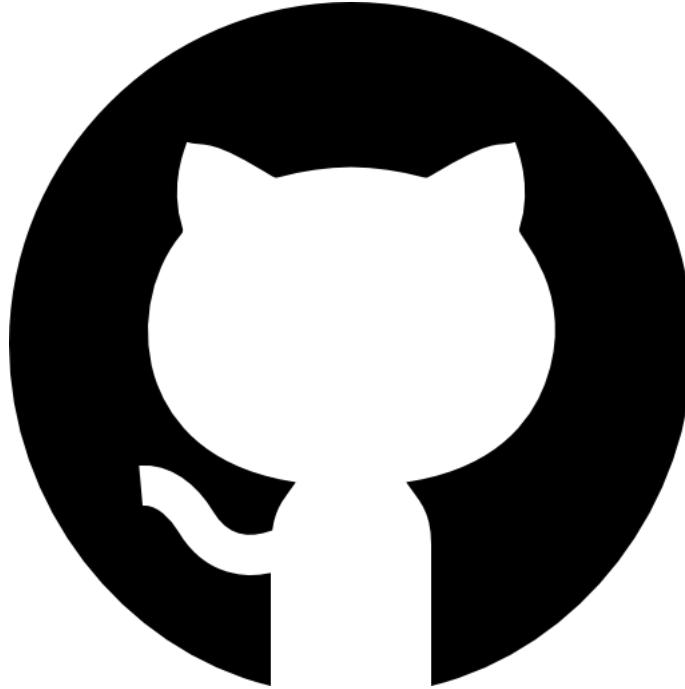
```
    font-size: 1.1rem;  
}
```

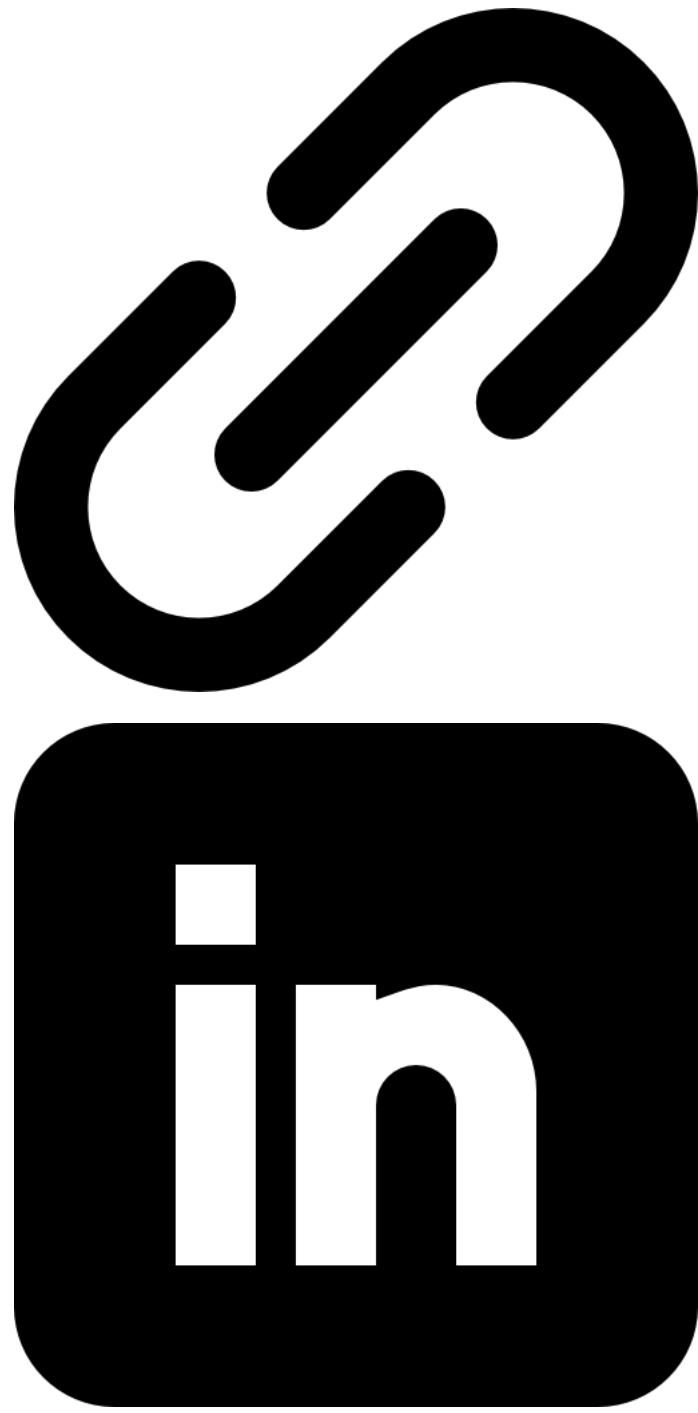
Una vez tenemos nuestros estilos globales definidos, vamos a estructurar nuestro proyecto mediante una serie de carpetas:

- components: contendrá los componentes reutilizables de nuestra aplicación
- pages: contendrá las páginas/vistas
- utils: contendrá funciones o métodos utiles para la aplicación
- data: contendrá los datos que vamos a consumir
- public/icons: contendrá iconos a modo de asset



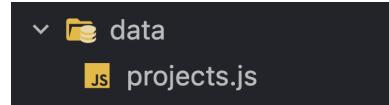
Dentro de la carpeta public/icons podremos almacenar los siguientes assets:





Datos

Dentro de la carpeta data, vamos a exportar una constante que almacenará toda la información de nuestros proyectos, ya que es la parte más importante de nuestro Portfolio. Para ello crearemos un array de objetos el cual almacenará el nombre del proyecto, una imagen, una descripción, el link al repositorio y el despliegue y una lista de las tecnologías utilizadas en los mismos.

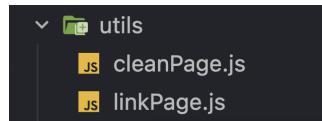


En este caso será información mockeada:

```
export const projects = [
  {
    title: "Primer proyecto",
    description: "Esta es la descripción de mi primer proyecto",
    image:
      "https://images.unsplash.com/photo-1498050108023-c5249f4df085?ixlib=rb-4.0.3&ixid=MnwxMjA3fDB8MHxzZWFyY2h8Mnx8d2VifGVufDB8fDB8fA%3D%3",
    tech: ["React", "HTML5", "CSS3", "JavaScript"],
    github: "http://github.com/myuser/proyecto1",
    link: "www.proyecto1.com",
  },
  {
    title: "Segundo proyecto",
    description: "Esta es la descripción de mi segundo proyecto",
    image:
      "https://images.unsplash.com/photo-166196111184-11317b40adb2?ixlib=rb-4.0.3&ixid=MnwxMjA3fDF8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8&auto=f",
    tech: ["React", "HTML5", "CSS3", "JavaScript"],
    github: "http://github.com/myuser/proyecto2",
    link: "www.proyecto1.com",
  },
  {
    title: "Tercer proyecto",
    description: "Esta es la descripción de mi tercer proyecto",
    image:
      "https://images.unsplash.com/photo-1669570094762-828f3dfa675?ixlib=rb-4.0.3&ixid=MnwxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8&auto=f",
    tech: ["React", "HTML5", "CSS3", "JavaScript"],
    github: "http://github.com/myuser/proyecto3",
    link: "www.proyecto1.com",
  },
  {
    title: "Cuarto proyecto",
    description: "Esta es la descripción de mi cuarto proyecto",
    image:
      "https://images.unsplash.com/photo-1664575198308-3959904fa430?ixlib=rb-4.0.3&ixid=MnwxMjA3fDF8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8&auto=f",
    tech: ["React", "HTML5", "CSS3", "JavaScript"],
    github: "http://github.com/myuser/proyecto4",
    link: "www.proyecto1.com",
  },
  {
    title: "Quinto proyecto",
    description: "Esta es la descripción de mi quinto proyecto",
    image:
      "https://images.unsplash.com/photo-1665686306574-1ace09918530?ixlib=rb-4.0.3&ixid=MnwxMjA3fDF8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8&auto=f",
    tech: ["React", "HTML5", "CSS3", "JavaScript"],
    github: "http://github.com/myuser/proyecto5",
    link: "www.proyecto1.com",
  },
];
```

Utils

En esta carpeta vamos a generar dos funciones sencillas que podremos utilizar en uno o varios componentes.



La primera de ellas será un limpiador de contenedores y nodos, ya que para darle vida a nuestra SPA tendremos que sustituir el contenido de los mismos por otros. Como esto se va a realizar en cada una de las páginas de nuestra aplicación, vamos a aislar esta funcionalidad en un fichero llamado :

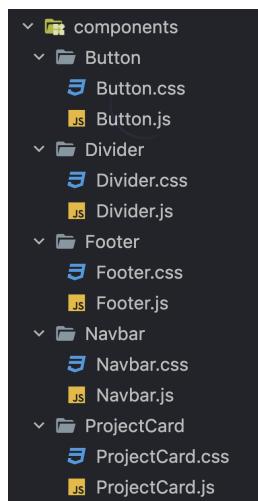
```
export const cleanPage = (container) => {
  container.innerHTML = "";
};
```

Por otro lado, necesitaremos una función que recoja cualquier link mediante id de nuestra aplicación y le dote de un addEventListener que, al hacer click, ejecute una función que pintará las diferentes páginas. Estas funciones que renderizan las páginas se las pasaremos también por argumento:

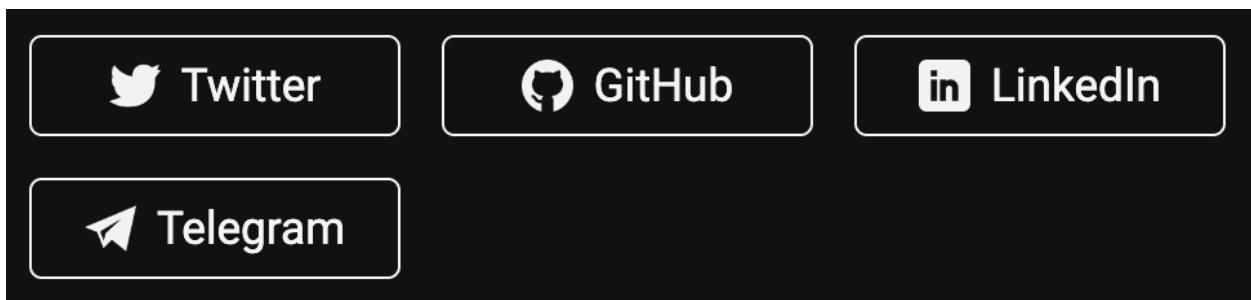
```
export const linkPage = (id, page) => {
  const link = document.querySelector(id);
  link.addEventListener("click", () => page());
};
```

Componentes

Antes de definir nuestras páginas, vamos a crear los componentes que vamos a reutilizar en nuestra aplicación. Para ello vamos a dividirlos en carpetas que contendrán tanto el script como sus estilos:



Empecemos con el componente **Button**, el cual reutilizaremos tantas veces como queramos en el footer de la aplicación:



Para ello vamos a configurar que pueda aceptar tanto un ícono como un texto, además de unos estilos personalizados:

```
import "./Button.css";

export const Button = (icon, text) => `
```

```
<button class="my-btn">
<img src=${icon} alt='${text} icon'>
<h4>${text}</h4>
</button>
`;
```

```
.my-btn {
background: none;
border: 1px solid var(--primary-dark);
display: flex;
justify-content: center;
align-items: center;
color: var(--primary-dark);
width: 9rem;
padding: 0.5rem;
gap: 0.5rem;
border-radius: 0.3rem;
transition: all 0.5s ease-in-out;
background-color: var(--bg-dark);
}

.my-btn > img {
height: 20px;
width: 20px;
filter: invert(95%);
}

.my-btn:hover {
background-color: var(--highlight);
}
```

Nuestro siguiente componente será un **Divider**, una simple línea que separará las secciones o los textos de nuestra aplicación. Como lo vamos a reutilizar más de una vez lo vamos a componentizar.

```
import "./Divider.css";

export const Divider = () => `
```

```
<span class="divider"></span>
`;
```

```
.divider {
border-bottom: 1px solid var(--divider);
height: 1px;
width: 100%;
display: block;
margin: 1rem 0;
}
```

El siguiente componente que vamos a crear es **Footer**, en el cual reutilizaremos el componente **Button** que hemos creado anteriormente para insertar los botones que estarán en él. Para ello lo importaremos a la hora de crearlo y lo ejecutaremos en el espacio donde queramos implementarlos:

```
import "./Footer.css";
import { Button } from "../Button/Button";

export const Footer = () => `
```

```
<h2>Contact</h2>
<div>
${Button("/icons/twitter.png", "Twitter")}
${Button("/icons/github.png", "GitHub")}
${Button("/icons/linkedin.png", "LinkedIn")}
${Button("/icons/telegram.png", "Telegram")}
</div>
';
```

```
footer {
  display: flex;
  justify-content: space-between;
  align-items: center;
  padding: 0 8rem;

  gap: 4rem;
  flex-wrap: wrap;
  margin: 2rem 0;
}

@media (max-width: 770px) {
  footer {
    padding: 0 2rem;
    gap: 2rem;
  }
}

footer > div {
  display: flex;
  gap: 1rem;
  flex-wrap: wrap;
}
```

En ciertos casos tendremos que hacer uso de medias-queries para adaptar a distintos dispositivos nuestra aplicación.

El siguiente componente que vamos a crear será **ProjectCard**, un componente que pintará cada uno de nuestros proyectos y su detalle.



Primer proyecto



Esta es la descripción de mi primer proyecto

React - HTML5 - CSS3 - JavaScript

Este componente aceptará por argumento un proyecto de nuestros datos y pintará en diversas etiquetas todos los datos.

```
import './ProjectCard.css';

export const ProjectCard = (project) => `

<div class="project-card">
<img src=${project.image} alt=${project.title}/>
<div class="header">
<h2>${project.title}</h2>
</div>
<a href=${project.github}>

</a>
<a href=${project.link}>

</a>
</div>
</div>

<div class="detail">
<p>${project.description}</p>
<p class="tech">${project.tech.join(" - ")})</p>
</div>
</div>
`;
```

```
.project-card {
  width: 100%;
  display: flex;
  flex-direction: column;
  gap: 1rem;
}
```

```

.project-card img {
  width: 100%;
  height: 250px;
  border-radius: 0.4rem;
  object-fit: cover;
}

.project-card > .header {
  display: flex;
  align-items: center;
  justify-content: space-between;
}

.project-card > .header > div {
  display: flex;
  gap: 1rem;
}

.project-card > .header > div > a > img {
  width: 20px;
  height: 20px;
  filter: invert(90%);
  transition: all 0.3s ease-in-out;
}

.project-card > .header > div > a > img:hover {
  transform: scale(1.1);
}

.project-card > .detail {
  display: flex;
  flex-direction: column;
  gap: 1rem;
}

.project-card > .detail > .tech {
  color: var(--highlight);
}

```

Por último, vamos a definir nuestro componente **Navbar**. Este componente contendrá dos funciones combinadas que utilizaremos más adelante que nos permitirá cambiar el tema y el texto del botón que cambiará el tema.



Además de esto, incluiremos los links que nos permitirán navegar entre las páginas con sus respectivos ids para, posteriormente, darle su escuchador de evento que almacenamos en utils:

```

import "./Navbar.css";

export const changeTheme = () => {
  const themeBtn = document.querySelector("#themeBtn");
  themeBtn.addEventListener("click", () => {
    document.body.classList.toggle("light");
    changeText();
  });
};

export const changeText = () => {
  const themeBtn = document.querySelector("#themeBtn");
  if (themeBtn.innerText === "*") {
    themeBtn.innerText = "☀️";
  } else {
    themeBtn.innerText = "*";
  }
};

export const Navbar = () => `
```

```

<nav>
<h2>Peter Parker</h2>
<ul>
  <li>
    <a href="#" id="homelink">Home</a>
  </li>
  <li>
    <a href="#" id="projectslink">Projects</a>
  </li>
  <li>
    <button id="themeBtn">*</button>
  </li>
</ul>
</nav>
';

```

```

nav {
  height: 10vh;
  display: flex;
  justify-content: space-between;
  align-items: center;
  padding: 0 8rem;
  flex-wrap: wrap;
}

@media (max-width: 770px) {
  nav {
    padding: 0 2rem;
  }
}

nav > ul {
  display: flex;
  gap: 3rem;
}

nav button {
  background: none;
  border: none;
  color: inherit;
  transform: scale(1.5);
}

nav a {
  transition: all 0.3s ease-in-out;
}

nav a:hover {
  color: var(--highlight);
}

```

Páginas

En nuestra carpeta **pages** vamos a almacenar las dos páginas que vamos a implementar en nuestra aplicación web, en este caso, una página **Home** y una página **Projects**.

Empezemos por **Home**:

Peter Parker

Home Projects ☀

Hey, I'm

Peter Parker

I'm a software engineer based in Toronto, Canada and also a communication and journalism student. I enjoy creating things that live on the internet, whether that be websites, applications, or anything in between. I have been freelancing for a year now while studying at the university and I've managed to gain a decent amount of experience and valuable knowledge from all different kinds of fields throughout my projects/work.

Say hi →

Contact

[Twitter](#) [GitHub](#) [LinkedIn](#) [Telegram](#)

Básicamente será una función que devuelve un template, vacía nuestro contenedor **main** del documento y lo rellenará con un template. Este vaciado del contenedor se efectuará gracias a nuestro util **cleanPage**.

```
import "./Home.css";
import { cleanPage } from "../../utils/cleanPage";

export const Home = () => {
  const main = document.querySelector("main");
  cleanPage(main);
  main.innerHTML = `
    <section class="home">
      <p>Hey, I'm</p>
      <h1>Peter Parker</h1>
      <p>I'm a software engineer based in Toronto, Canada and also a communication and journalism student. I enjoy creating things that live on the internet, whether that be websites, applications, or anything in between. I have been freelancing for a year now while studying at the university and I've managed to gain a decent amount of experience and valuable knowledge from all different kinds of fields throughout my projects/work.</p>
      <a href="mailto:peterparker@gmail.com">Say hi →</a>
    </section>`;
}
```

```
.home {
  padding: 0 8rem;
  height: 80vh;
  display: flex;
  flex-direction: column;
  justify-content: center;
  align-items: flex-start;
  gap: 3rem;
}

@media (max-width: 700px) {
  .home {
    padding: 0 2rem;
  }
}

.home > h1 {
  font-size: 5.5rem;
  background: linear-gradient(to right, #9845e8 0%, #33d2ff 50%, #dd5789 100%);
  -webkit-background-clip: text;
  background-clip: text;
  -webkit-text-fill-color: transparent;
```

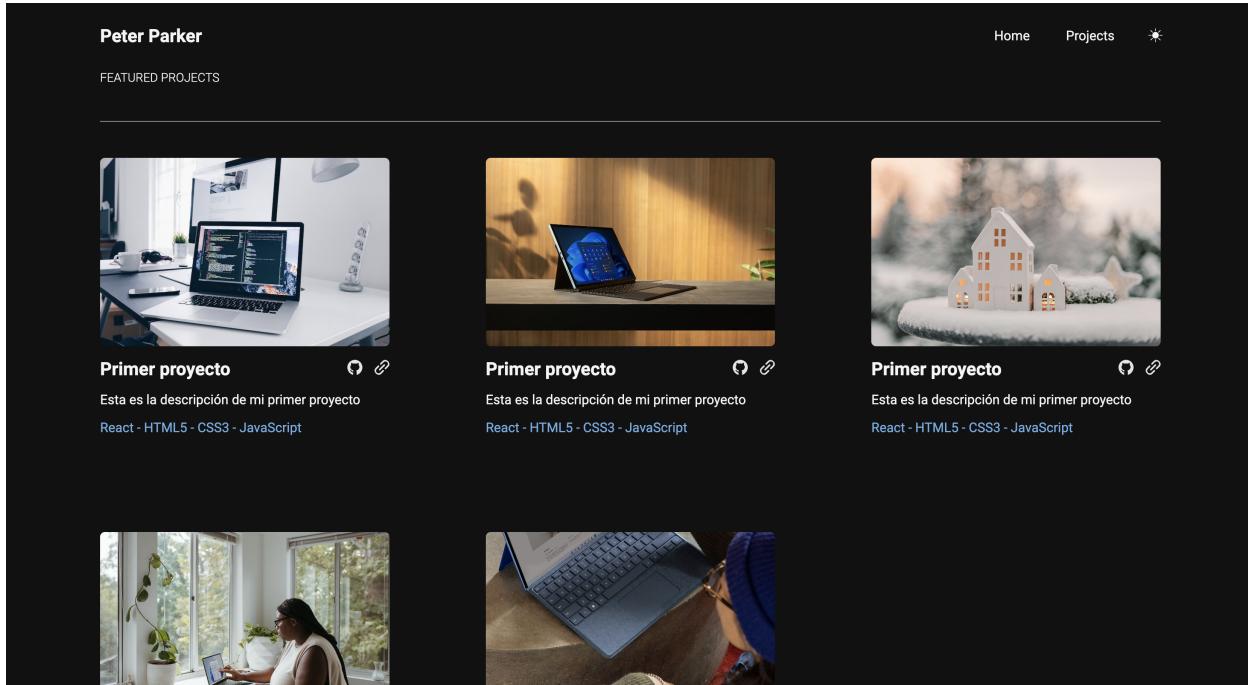
```

}

.home > a {
  font-size: 1.5rem;
}

```

Y la página que nos queda para completar todas las piezas de nuestra aplicación, la página **Projects**:



En este componente página utilizaremos también el útil **cleanPage**, importaremos los datos de projects almacenados en **data**, el componente **ProjectCard** al que le pasaremos cada uno de los proyectos a través de un bucle, y el **Divider** para implementar la línea decorativa de la parte superior entre el título y la propia galería. Estas cartas las insertaremos manipulando el DOM:

```

import "./Projects.css";
import { cleanPage } from "../../../../utils/cleanPage";
import { projects } from "../../../../data/projects";
import { ProjectCard } from "../../../../components/ProjectCard/ProjectCard";
import { Divider } from "../../../../components/Divider/Divider";

export const Projects = () => {
  const main = document.querySelector("main");
  cleanPage(main);
  main.innerHTML = `
    <section class="projects">
      <h2>Featured Projects</h2>
      ${Divider()}
      <div class="projects-container"></div>
    </section>`;
  const container = document.querySelector(".projects-container");
  for (const project of projects) {
    const figure = document.createElement("figure");
    figure.innerHTML = ProjectCard(project);
    container.appendChild(figure);
  }
};

```

```

.projects {
  padding: 0 8rem;
  min-height: 80vh;
  padding-bottom: 4rem;
  display: flex;
  flex-direction: column;
  gap: 2rem;
  margin-bottom: 2rem;
}

@media (max-width: 770px) {
  .projects {
    padding: 0 2rem;
  }
}

.projects > h2 {
  font-weight: 100;
  text-transform: uppercase;
  font-size: 1rem;
}

.projects-container {
  display: grid;
  grid-template-columns: repeat(auto-fill, minmax(300px, 1fr));
  gap: 8rem;
}

@media (max-width: 770px) {
  .projects-container {
    grid-template-columns: repeat(auto-fill, minmax(200px, 1fr));
  }
}

```

Ejecutando componentes, páginas y escuchadores de eventos

Lo único que nos queda por ejecutar en nuestro fichero **main.js**, el cual tendrá todos los componentes, páginas, útiles y funciones implementadas ejecutándose en un orden concreto y añadiéndole funcionalidad a la aplicación.

Una vez importadas todas las piezas pintaremos dentro de header el **Navbar** y en footer el componente **Footer**.

Una vez estén pintadas en el documento, implementaremos a través de la función **linkPage** la orden de pintar la página Home y Projects.

Por defecto pintaremos la página **Home** y ejecutaremos **changeTheme** para dotar de funcionalidad al **Navbar**. Esto se realiza en este punto porque ya tendríamos renderizado el navegador en el documento.

Para terminar, le añadimos justo antes del footer otro componente **Divider** por motivos estéticos.

```

import "./style.css";
import { changeTheme } from "./components/Navbar/Navbar";
import { linkPage } from "./utils/linkPage";
import { Navbar } from "./components/Navbar/Navbar";
import { Footer } from "./components/Footer/Footer";
import { Home } from "./pages/Home/Home";
import { Projects } from "./pages/Projects/Projects";
import { Divider } from "./components/Divider/Divider";

const header = document.querySelector("header");
header.innerHTML = Navbar();
const footer = document.querySelector("footer");
footer.innerHTML = Footer();

linkPage("#homelink", Home);

```

```
linkPage("#projectslink", Projects);  
Home();  
changeTheme();  
footer.insertAdjacentHTML("beforebegin", Divider());
```

Reto

Una vez completada nuestra aplicación, es vuestro turno de completar los siguientes pasos y cumplir el reto propuesto:

- Buscad inspiración e implementar vuestros propios estilos.
- Añadir información real de los proyectos que vais realizando durante el bootcamp.
- Componentizar más los elementos de la aplicación (título, elementos de navegación, imágenes, etc...).
- Añadir una página adicional, ya sea un **About**, un blog o una página dedicada al contacto.
- Desplegar la página en algún servicio gratuito de despliegue, por ejemplo Vercel.