

Evidencia de Producto

1. Creación de pagina en React:

Para la realización de esta pagina con react se ingresa el siguiente comando `npx create-react-app` en nuestra terminal.

Luego se creo la carpeta del proyecto, dentro de el proyecto se crea una nueva carpeta llamada components la cual contiene los componentes header, body, y footer.js .

Header.js:

Aqui creamos un encabezado de la pagina web, que incluye un menú de navegación con tres opciones .

```
1  import React from 'react';/*Esta línea importa la biblioteca React y
2  hace que esté disponible en este archivo.*/
3
4
5
6  function Header() { /*Definimos una función que se convierte en un componente*/
7    return ( /* la función nos devuelve los elementos de la interfaz*/
8
9    /*Apartir de aqui definimos la estructura de nuestro
10     * header que contiene un div el cual tiene una clase,
11     * y dentro del div tenemos una etiqueta nav tambien con su clase, dentro de esta
12     * encontramos una lista que muestra el menu de navegación de nuestra página.
13     */
14     <header>
15       <div className='header-container'>
16         <nav class="menu">
17           <ul>
18             <li>Inicio</li>
19             <li>Acerca de</li>
20             <li>contacto</li>
21           </ul>
22         </nav>
23       </div>
24     </header>
25   );
26 }
27 /*Aqui al final definos la exportación de nuestro
28 componente header para usarlo en nuestra app */
29 export default Header;
```

Body.js:

Aqui creamos el cuerpo de la pagina web, que incluye una estructura basica de un titulo y un parrafo .

```
1 import React from 'react';/*Esta línea importa la biblioteca React y
2 hace que esté disponible en este archivo.*/
3
4 function Body() { /*Aqui definimos una función llamada body que al igual que el header
5 se convierte en un componente*/
6   return ( /*la función nos devuelve los elementos de la interfaz*/
7
8     /*Aqui creamos un contenedor que contiene un titulo y un parrafo dentro de un div
9     que tiene una clase llamada container que nos permite hacer mención de nuestro div
10    en la app*/
11     <div class="container">
12       <h1>Bienvenido, esta es una pagina web minimalista hecha con React</h1>
13       <p>Este es el contenido de la Aplicación en React</p>
14     </div>
15   );
16 }
17 /*Aqui al final definos la exportación de nuestro
18 componente Body para usarlo en nuestra app*/
19 export default Body;
```

Footer.js:

Creamos el pie de pagina , que contiene tres simples parrafos.

```
1  import React from 'react'; /*Esta línea importa la biblioteca React y
2  hace que esté disponible en este archivo.*/
3  import '../css/footer.css'; /*Esta importa el css de para una mejor apariencia*/
4
5  function Footer() { /*Aqui definimos una función llamada footer que
6  al igual que el header se convierte en un componente*/
7
8      return (/*Devolvemos el contenido de la interfaz */
9
10         /*Aqui creamos un pie de pagina estructurado dentro de su etiqueta footer y dos div
11         con sus clases que nos permiten ser mencionados en el css para darle una mejor apariencia,
12         y dentro de los div creamos unos simples parrafos con información basica */
13         <footer>
14             <div class="footer-container">
15                 <div class="footer-info">
16                     <p>&copy; 2023 Framework React</p>
17                     <p>Contactanos: +32344343</p>
18                     <p>Correo: proyecto@react.com</p>
19                 </div>
20             </div>
21         </footer>
22     );
23 }
24
25 /*Exportamos el Footer para poder usarlo en toda nuestra app */
26 export default Footer;
27
```

App.js:

Por ultimo en el archivo principal hacemos mención de los componentes que deseamos ver en nuestra página.

```
1 import React from 'react'; // Importamos la biblioteca React, necesaria para trabajar con React.
2 import './App.css'; // Importamos un archivo CSS llamado 'App.css' para aplicar estilos a nuestra aplicación.
3 import Header from './components/Header'; // Importamos el componente 'Header' desde el archivo header.js
4 import Body from './components/Body'; // Importamos el componente 'Body' desde el archivo 'Body.js'
5 import Footer from './components/Footer'; // Importamos el componente 'Footer' desde el archivo 'Footer.js'.
6 import './App.css';
7
8 function App() { /* Definimos una función llamada 'App' que representa el componente
9 principal de nuestra aplicación.*/
10   return ( /* Iniciamos el cuerpo de nuestra función, que contendrá el código
11     que describe la estructura de la interfaz de usuario.*/
12     /*Creamos un contenedor 'div' con una clase llamada "App". Esta clase se usa
13     para aplicar estilos CSS desde el archivo 'App.css'.*/
14
15     // "Header" Insertamos el componente 'Header' en nuestra aplicación.
16     // "Body" Insertamos el componente 'Body'.
17     // "Footer" Insertamos el componente 'Footer'.
18     <div className="App">
19       <Header />
20       <Body />
21       <Footer />
22     </div>
23   );
24 }
25 //Exportamos el componente app pero que puede ser utilizado en otros archivos de la app.
26 export default App;
```

2. Creación de página en Angular:

Lo primero que se debe realizar es instalar angular con el comando: “npm install –g @angular/cli”. Luego de la instalación se realiza el siguiente comando para crear el proyecto “ng new my-angular-app” .

Luego dentro del proyecto va a estar presente una carpeta la cual va a contener toda la estructura de nuestro proyecto llamada “app” en la cual vamos a crear

nuestros componentes header, body y footer , la diferencia es que aquí para crear un componente debemos realizar el comando: “ng generate component nombre-del-componente”, este nos crea una carpeta dentro de la carpeta “app” que va a contener toda su estructura necesaria para que pueda funcionar, esta contiene un archivo html , css y dos archivos logicos.

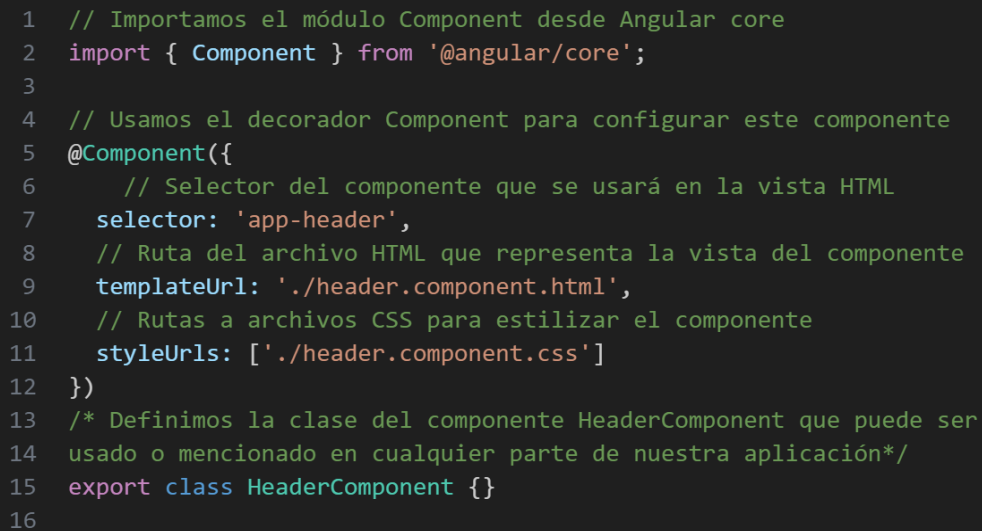
Header:

Ahora para la creación de nuestra estructura html que va a contener nuestro encabezado hacemos mención de esta dentro del archivo “header.component.html” ya definido.

```
1
2 <!--/**Apartir de aquí definimos la estructura de nuestro
3  * header que contiene un div el cual tiene una clase,
4  * y dentro del div tenemos una etiqueta nav tambien con su clase, dentro de esta
5  * encontramos una lista que muestra una imagen como logo y el menu de navegación de nuestra página.
6  */-->
7 <header>
8   <div className='header-container'>
9     <nav class="menu">
10      <ul>
11        <li class="logo"></li>
12        <li>Inicio</li>
13        <li>Acerca de</li>
14        <li>contacto</li>
15      </ul>
16    </nav>
17  </div>
18 </header>
```

header.component.ts:


Para la funcionalidad y visibilidad de este código debemos revisar que en el archivo “header.component” se haga mención de los archivos dentro de “@Component que es un decorador y en este pasamos las propiedades de nuestro componente.



```
1 // Importamos el módulo Component desde Angular core
2 import { Component } from '@angular/core';
3
4 // Usamos el decorador Component para configurar este componente
5 @Component({
6   // Selector del componente que se usará en la vista HTML
7   selector: 'app-header',
8   // Ruta del archivo HTML que representa la vista del componente
9   templateUrl: './header.component.html',
10  // Rutas a archivos CSS para estilizar el componente
11  styleUrls: ['./header.component.css']
12 })
13 /* Definimos la clase del componente HeaderComponent que puede ser
14 usado o mencionado en cualquier parte de nuestra aplicación*/
15 export class HeaderComponent {}
16
```

Body:

En el cuerpo de la página realizados de igual forma su estructura html en el archivo “body.component.html”.




```

1  <!--Aquí creamos un contenedor que contiene un título y un párrafo
2     dentro de un div que tiene una clase llamada container que nos
3     permite hacer mención de nuestro div en la app-->
4  <div class="container">
5     <h1>Bienvenido, esta es una página web minimalista hecha con Angular</h1>
6     <p>Este es el contenido de la Aplicación en Angular</p>
7  </div>
8

```

“body.component.ts”:

Para la funcionalidad y visibilidad de este código debemos revisar que en el archivo “body.component” se haga mención de los archivos dentro de “@Component que es un decorador y en este pasamos las propiedades de nuestro componente.



```

1  // Importamos el módulo Component desde Angular core
2  import { Component } from '@angular/core';
3
4  // Usamos el decorador Component para configurar este componente
5  @Component({
6     // Selector del componente que se usará en la vista HTML
7     selector: 'app-body',
8     // Ruta del archivo HTML que representa la vista del componente
9     templateUrl: './body.component.html',
10    // Rutas a archivos CSS para estilizar el componente
11    styleUrls: ['./body.component.css']
12  })
13  /* Definimos la clase del componente BodyComponent que puede ser
14     usada o mencionada en cualquier parte de nuestra aplicación*/
15  export class BodyComponent {
16
17  }
18

```

Footer:

En este también definimos su estructura html para el pie de página en el archivo “footer.component.html”.

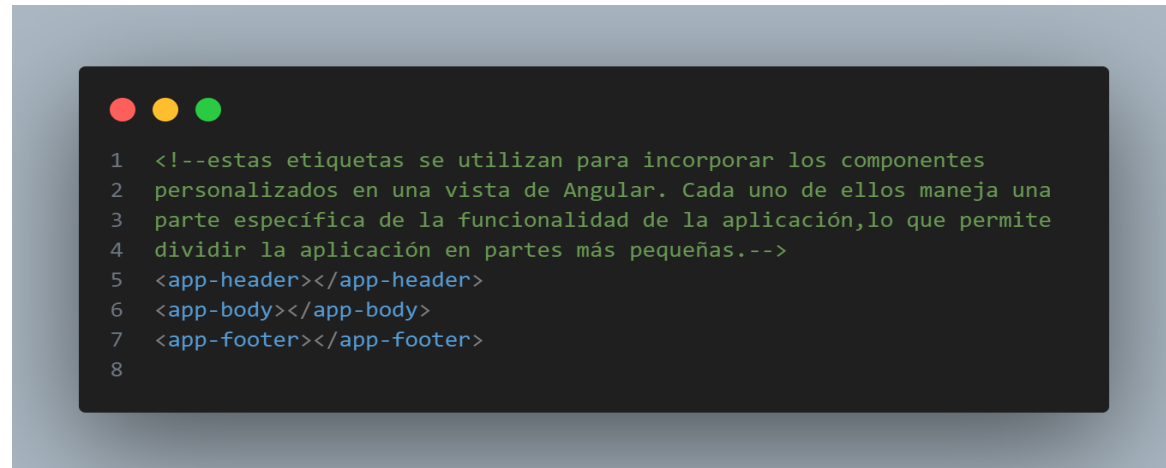
```
1  <!--Aquí creamos un pie de pagina estructurado dentro de su etiqueta footer y dos div
2   con sus clases que les permiten ser mencionados en el css para darle una mejor apariencia,
3   y dentro de los div creamos tres simples parrafos con información básica -->
4  <footer>
5    <div class="footer-container">
6      <div class="footer-info">
7        <p>&copy; 2023 Framework Angular</p>
8        <p>Contactanos: +32344343</p>
9        <p>Correo: proyecto@angular.com</p>
10     </div>
11   </div>
12 </footer>
13
```

Footer.component.html: Para la funcionalidad y visibilidad de este código debemos revisar que en el archivo “footer.component” se haga mención de los archivos dentro de “@Component que es un decorador y en este pasamos las propiedades de nuestro componente.

```
1  // Importamos el módulo Component desde Angular core
2  import { Component } from '@angular/core';
3
4  // Usamos el decorador Component para configurar este componente
5  @Component({
6    // Selector del componente que se usará en la vista HTML
7    selector: 'app-footer',
8    // Ruta del archivo HTML que representa la vista del componente
9    templateUrl: './footer.component.html',
10   // Rutas a archivos CSS para estilizar el componente
11   styleUrls: ['./footer.component.css']
12 })
13 /* Definimos la clase del componente FooterComponent que puede ser
14  usado o mencionado en cualquier parte de nuestra aplicación*/
15 export class FooterComponent {
16
17 }
18
```


App.component.html:

Por ultimo en el archivo principal hacemos mención de los componentes que deseamos ver en nuestra página.



3. Creación página en Vue:

Para la realización de esta página se debe primero intalar vue CLI con el siguiente comando “npm install -g @vue/cli” Luego de la instalación se realiza el siguiente comando para crear el proyecto o aplicación “vue create my-vue-app”.

Luego dentro del proyecto va a estar presente una carpeta la cual va a tener nuestros componentes del proyecto llamada “components” en la cual vamos a crear nuestros componentes header, body y footer , la diferencia es que aquí al crear un nuevo componente se pone “.vue” y dentro de este archivo podemos definir toda la estructura de nuestro componente como el html, su js y su css, todo en un solo archivo.

Header.vue:

Este código representa el componente de encabezado, que incluye su estructura html, configuración y estilos específicos para el componente. El nombre del componente es 'PageHeader'.

```
1 <template><!--Define la estructura del componente-->
2
3 <!--Apartir de aqui definimos la estructura de nuestro
4  * header que contiene un div el cual tiene una clase,
5  * y dentro del div tenemos una etiqueta nav tambien con su clase, dentro de esta
6  * encontramos una lista que muestra el menu de navegación de nuestra página.-->
7  <header>
8    <div class="header-container">
9      <nav class="menu">
10        <ul>
11          <li>Inicio</li>
12          <li>Acerca de</li>
13          <li>Contacto</li>
14        </ul>
15      </nav>
16    </div>
17  </header>
18 </template>
19
20 <script>/*Contiene la configuración y la lógica del componente.
21 Aquí, se establece el nombre del componente como 'PageHeader',
22 que se utilizará para identificarlo en la aplicación.*/
23 export default {
24   name: 'PageHeader',
25 }
26 </script>
27
28 <!-- Estilos específicos para este componente -->
29 <style scoped>
30 header {
31   background-color: #302e2e; /* Color de fondo del header */
32   color: #fff; /* Color del texto */
33   padding: 1rem; /* Espaciado interno del header */
34 }
```

Body.vue:

Este código representa el cuerpo de la página , que incluye su estructura html, configuración y estilos específicos para el componente. El nombre del componente es 'PageBody'.

```
1 <template><!--Define la estructura del componente-->
2
3 <!--Aquí creamos un contenedor que contiene un titulo y un parrafo
4 dentro de un div que tiene una clase llamada container que nos permite
5 hacer mención de nuestro div en la app-->
6   <div class="container">
7     <h1>Bienvenido, esta es una pagina web minimalista hecha con Vue</h1>
8     <p>Este es el contenido de la Aplicación en Vue</p>
9   </div>
10 </template>
11
12 <script>/*Contiene la configuración y la lógica del componente.
13 Aquí, se establece el nombre del componente como 'PageBody',
14 que se utilizará para identificarlo en la aplicación.*/
15 export default {
16   name: 'PageBody',
17 }
18 </script>
19
20 <!--Estilos específicos para este componente, si es necesario-->
21 <style scoped>
22 .container {
23   background: url(../assets/Vue.png);/*Agrega una imagen de fondo
24   en el contenedor */
25   padding: 18rem;
26 }
```

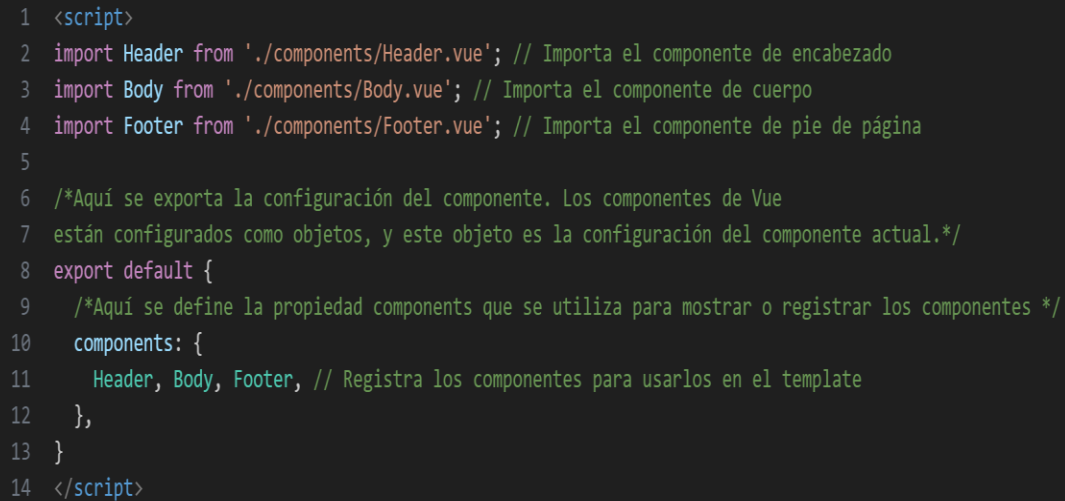
Footer.vue:

Este código representa el pie de página , que incluye su estructura html, configuración y estilos específicos para el componente. El nombre del componente es 'PageFooter'.

```
1  <template><!--Define la estrucutra del componente-->
2
3  <!--Aquí creamos un pie de pagina estructurado dentro de su etiqueta
4  footer y dos div con sus clases que les permiten ser mencionados en el css
5  para darle una mejor apariencia, y dentro de los div  creamos unos simples
6  parrafos con información basica -->
7    <footer>
8      <div class="footer-container">
9        <div class="footer-content">
10       </div>
11       <div class="footer-info">
12         <p>&copy; 2023 Framework Vue</p>
13         <p>Contactanos: +32344343</p>
14         <p>Correo: proyecto@vue.com</p>
15       </div>
16     </div>
17   </footer>
18 </template>
19
20 <script>/*Contiene la configuración y la lógica del componente.
21 Aquí, se establece el nombre del componente como 'PageFooter',
22 que se utilizará para identificarlo en la aplicación.*/
23 export default {
24   name: 'PageFooter',
25 }
26 </script>
27
28 <!--Estilos específicos para este componente, si es necesario-->
29 <style scoped>
30 footer {
31   background-color: #318052; /* Color de fondo del footer */
32   color: #fff; /* Color del texto */
33   padding: 5rem; /* Espaciado interno del footer */
34 }
```

App.vue:

Por ultimo importamos los componentes “header”, “body” y “footer” y los registra para su uso en el componente principal.

A screenshot of a code editor with a dark background and light-colored text. The code is written in a syntax-highlighted style. At the top left of the editor window, there are three colored circles (red, yellow, green) representing window control buttons. The code is as follows:

```
1 <script>
2 import Header from './components/Header.vue'; // Importa el componente de encabezado
3 import Body from './components/Body.vue'; // Importa el componente de cuerpo
4 import Footer from './components/Footer.vue'; // Importa el componente de pie de página
5
6 /*Aquí se exporta la configuración del componente. Los componentes de Vue
7 están configurados como objetos, y este objeto es la configuración del componente actual.*/
8 export default {
9   /*Aquí se define la propiedad components que se utiliza para mostrar o registrar los componentes */
10  components: {
11    Header, Body, Footer, // Registra los componentes para usarlos en el template
12  },
13 }
14 </script>
```