

UNIVERSIDAD MARIANO GÁLVEZ DE GUATEMALA

FACULTAD DE INGENIERÍA EN SISTEMAS

DESARROLLO WEB

ING. JOSÉ MIGUEL VILLATORO HIDALGO



MANUAL TÉCNICO

PROYECTO 2

KELVIN JOSÉ GÓMEZ MORALRES

9490-19-480

LESTER HAROLDO BLANCO MELENDRES

9490-19-5517

GUATEMALA, 24 DE OCTUBRE DE 2023

Tabla de contenidos

1.	Esquema conceptual de componentes.....	4
1.1.	Usuario	4
1.2.	Admin.....	5
2.	Descripción de componentes	5
2.1.	AuthContext	5
2.2.	AvisoError.....	6
2.3.	AvisoExitoso	6
2.4.	AvisoLogin.....	6
2.5.	CarritoModal	6
2.6.	ComprasHistorial	7
2.7.	Datos.....	7
2.8.	DatosCompras	7
2.9.	Footer	8
2.10.	Header.....	8
2.11.	Home	8
2.12.	Login.....	8
2.13.	Perfil	9
2.14.	ProductoDetalle	9
2.15.	ProductoDetalleAdmin	9
2.16.	Productos	10
2.17.	ProductosAdmin	10
2.18.	ProductosNuevo.....	10
2.19.	Registro.....	10
2.20.	SelectorCategorías	11
3.	Uso de Typescript	11
3.1.	Archivos usados	11
3.1.1.	AuthContext.tsx y AuthContextTypes.ts	11
3.1.2.	AvisoExitoso.tsx / AvisoError.tsx y AvisoErrorInterface.ts /Aviso ExitosoInterface.ts	12
3.1.3.	AvisoLogin.tsx.....	14
3.1.4.	SelectorCategorías.tsx.....	14

4.	Uso de SCSS	15
4.1.	CarritoModal.scss.....	15
4.2.	Datos.scss y DatosCompras.scss	15
4.3.	Header.scss	16
4.4.	Perfil.scss.....	16
4.5.	ProductoDetalle.scss	17
4.6.	ProductoDetalleAdmin.scss y ProductosNuevo.scss	17
4.7.	Registro.scss	18
5.	Enlaces de los proyectos	19
5.1.	API's en Render.com	19
5.2.	Proyecto React en Netlify.com.....	19

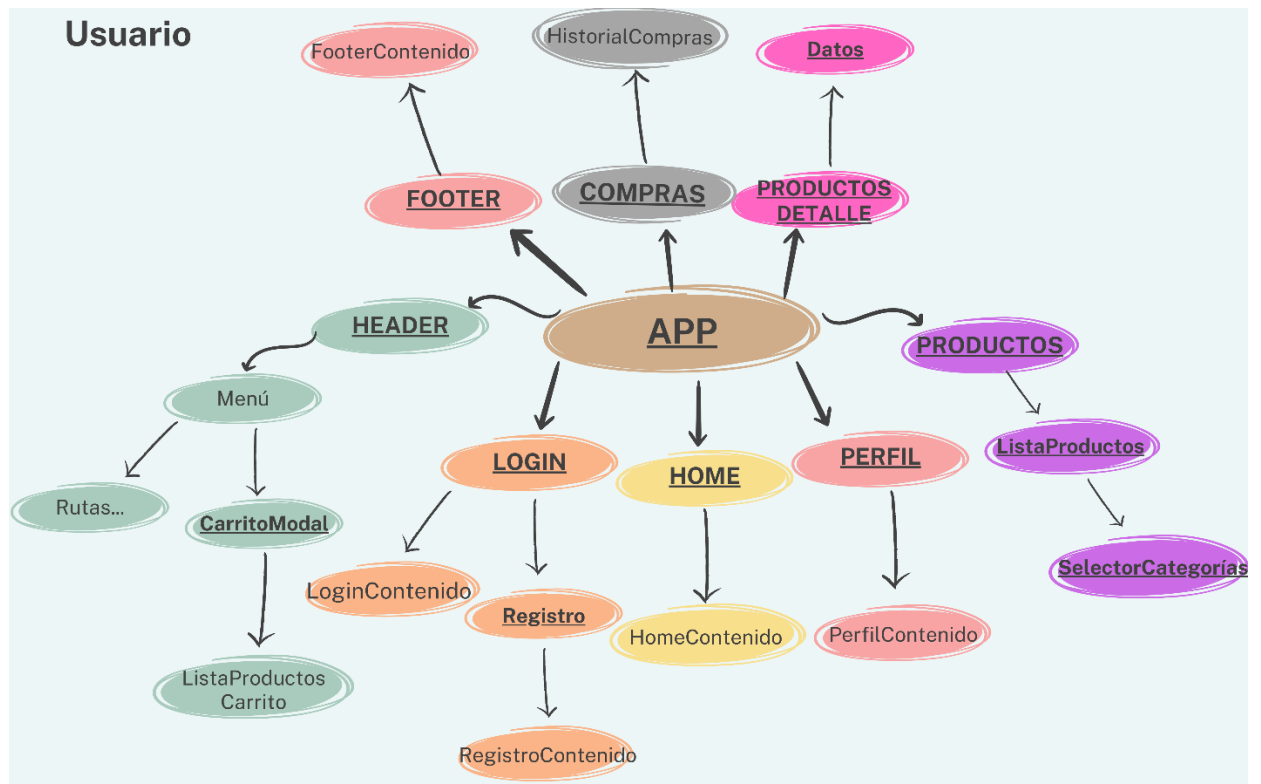
1. Esquema conceptual de componentes

Este tipo de esquemas nos muestran una nueva vista hacia nuestros componentes, donde se puede apreciar desde que lugar se puede acceder a ellos y encontrar su contenido para reutilizarlos.

En este caso nuestros componentes, fueron separados en esquemas diferentes, ya que en el Login se valida el tipo de usuario y nos permite mostrar distintos componentes en rutas distintas para delimitar aún más las acciones por medio de la decodificación del token.

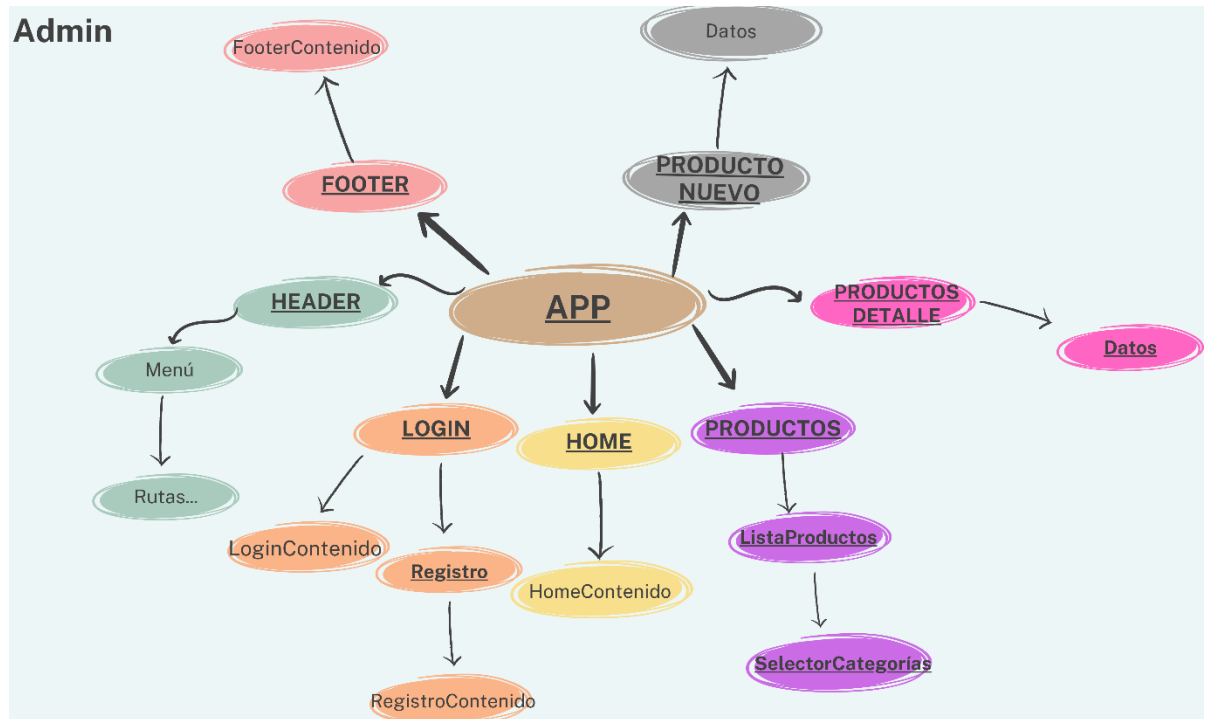
1.1. Usuario

Estos serán los componentes y su estructura, siempre teniendo en cuenta el AuthContext que es el contexto utilizado para el manejo de sesión.



1.2. Admin

Estos son los componentes que se le presentan al administrador para que pueda gestionar los productos del listado general.



2. Descripción de componentes

2.1. AuthContext

Ruta de los archivos: './src/AuthContext/' incluye el AuthContext.tsx y AuthContextTypes.ts

Estos archivos brindan el provider de autenticación, mantienen y crean un contexto el cual está disponible en toda la aplicación y manejan las acciones de 'Login' y 'Logout' así como sus distintas propiedades que validan las sesiones.

2.2. AvisoError

Ruta de los archivos: './src/AvisoError/' incluye el AvisoError.tsx y AvisoErrorTypes.ts

Este componente es un modal el cual se activa en las situaciones en las que nos retorna un mensaje de error en cualquier parte de la aplicación, recibe como props el mensaje y sus handles de mostrar y ocultar.

2.3. AvisoExitoso

Ruta de los archivos: './src/AvisoExitoso/' incluye el AvisoExitoso.tsx y AvisoExitosoInterface.ts

Este componente es un modal el cual se activa en las situaciones en las que nos retorna un mensaje de éxito en cualquier parte de la aplicación, recibe como props el mensaje y sus handles de mostrar y ocultar.

2.4. AvisoLogin

Ruta de los archivos: './src/AvisoLogin/' incluye el AvisoLogin.tsx

Este componente nos muestra un mensaje de que debemos iniciar sesión para acceder a las rutas y recursos de la aplicación.

2.5. CarritoModal

Ruta de los archivos: './src/CarritoModal/' incluye el CarritoModal.js y CarritoModal.scss

Este componente es un modal el cual se activa en las situaciones en las que presionamos el botón de ‘Ver Carrito’ en el header, si somos un usuario. Este modal tiene handles de mostrar y ocultar, así como la acción de poder comprar productos del carrito.

2.6. ComprasHistorial

Ruta de los archivos: ‘./src/ComprasHistorial/’ incluye el ComprasHistorial.js

Este componente contiene otros componentes y que al invocarlos y renderizarlos, nos muestran la información de las compras realizadas por el usuario, renderiza el producto, su imagen en una lista, así como recuperar los detalles más importantes de una compra.

2.7. Datos

Ruta de los archivos: ‘./src/Datos/’ incluye el Datos.js y Datos.scss

Este componente contiene la información de los productos que serán mostrados a el usuario, renderiza el producto, su imagen, así como recuperar los detalles más importantes que puedan servirle al cliente a visualizar el producto antes de agregarlo al carrito.

2.8. DatosCompras

Ruta de los archivos: ‘./src/DatosCompras/’ incluye el DatosCompras.js y DatosCompras.scss

Este componente es el encargado de recibir los props de ComprasHistorial para que pueda renderizar los datos de las compras realizadas.

2.9. Footer

Ruta de los archivos: './src/Footer/' incluye el Footer.js

Este componente se renderiza durante toda la aplicación, en este caso incluye la información de los desarrolladores del programa.

2.10. Header

Ruta de los archivos: './src/Header/' incluye el Header.js y Header.scss

Este componente se renderiza durante toda la aplicación, en este caso incluye las diferentes opciones a las que según que tipo de usuario sea, puede acceder y tiene permisos específicos para navegar, recibir y enviar peticiones.

2.11. Home

Ruta de los archivos: './src/Home/' incluye el Home.js

Este componente se renderiza como el primer componente que ve el usuario al momento de ingresar las credenciales correctas y generar un token. Existe una versión diferente para cada tipo de usuario.

2.12. Login

Ruta de los archivos: './src/Login/' incluye el Login.js

Este componente es el primero que ve el usuario al ingresar a la url correspondiente en donde se esté ejecutando el proyecto2, contiene la opción de ingresar las credenciales o caso contrario, no tener una cuenta, se puede registrar cualquier persona.

2.13. Perfil

Ruta de los archivos: './src/Perfil/' incluye el Perfil.js y Perfil.scss.

Este componente se encarga de que todos los usuarios puedan ver su información personal, puedan modificarla en cualquier momento o si fuese el caso, también pueden eliminar su cuenta de forma permanente y con ello la invalidación de todos sus permisos y medios de acceso.

2.14. ProductoDetalle

Ruta de los archivos: './src/ProductoDetalle/' incluye el ProductoDetalle.js y ProductoDetalle.scss.

Este componente renderiza los detalles del producto que haya elegido el usuario dentro de toda la lista de producto.

2.15. ProductoDetalleAdmin

Ruta de los archivos: './src/ProductoDetalleAdmin/' incluye el ProductoDetalleAdmin.js y ProductoDetalleAdmin.scss.

Este componente renderiza los detalles del producto que haya elegido el administrador dentro de toda la lista de productos y permitirá editar todos sus detalles.

2.16. Productos

Ruta de los archivos: './src/Productos/' incluye el Productos.js

Este componente renderiza el listado de los productos específicos para el usuario, donde se encontrará también un componente para filtrar por categoría.

2.17. ProductosAdmin

Ruta de los archivos: './src/ProductosAdmin/' incluye el ProductosAdmin.js

Este componente renderiza el listado de los productos específicos para el administrador, donde se encontrará también un componente para filtrar por categoría y un botón que le permitirá ingresar nuevos productos.

2.18. ProductosNuevo

Ruta de los archivos: './src/ProductosNuevo/' incluye el ProductosNuevo.js y ProductosNuevo.scss

Este componente se muestra cuando el administrador da click al botón para redireccionarlo a esa parte, acá el administrador podrá ingresar un producto desde 0 y automáticamente después de añadirlo, los clientes podrán visualizarlo y manipularlo, ya sea agregándolo a los carritos, comprándolo o simplemente consultado sus detalles.

2.19. Registro

Ruta de los archivos: './src/Registro/' incluye el Registro.js y Registro.scss

Este componente permite a los usuarios que aún no tengan una cuenta, crearla de acuerdo con sus datos personales, tiene validaciones en todos sus campos para que únicamente se puedan colocar valores correctos.

2.20. SelectorCategorías

Ruta de los archivos: './src/SelectorCategorías/' incluye el SelectorCategorías.tsx

Este componente permite a los usuarios y administradores que mediante un elemento select separado en un componente, puedan en el listado de productos correspondiente, hacer las filtraciones necesarias, puede hacer ese filtrado por cada categoría o elegir ver todos los elementos.

3. Uso de Typescript

3.1. Archivos usados

3.1.1. AuthContext.tsx y AuthContextTypes.ts

Se utilizo en acá, con un interface el cual nos ayudará a pasarle el tipo de dato que necesitamos en específico para verificar la autenticación y pasarle los props al contexto correspondiente.

```
TS AuthContextTypes.ts X
proyecto2 > src > AuthContext > TS AuthContextTypes.ts > ...
1  import { ReactNode } from 'react';
2
3  export interface AuthState {
4    | isAuthenticated: boolean;
5  }
6
7  export type AuthAction = { type: 'LOGIN' } | { type: 'LOGOUT' };
8
9  export interface AuthContextProps {
10    state: AuthState;
11    dispatch: React.Dispatch<AuthAction>;
12    usuario: any;
13    setUsuario: React.Dispatch<any>;
14    token: any;
15    setToken: React.Dispatch<any>;
16    rol: any;
17    setRol: React.Dispatch<any>;
18    dpi: any;
19    setDpi: React.Dispatch<any>;
20  }
21
22  export interface AuthProviderProps {
23    | children: ReactNode;
24  }
25
```

El contexto al que se lo pasamos es AuthContext.tsx. Acá algunas imágenes de los props y las demás propiedades.

```
import React, { createContext, useReducer, useContext, useState } from 'react';
import { AuthState, AuthAction, AuthContextProps, AuthProviderProps } from './AuthContextTypes';

const AuthContext = createContext<AuthContextProps | undefined>(undefined);

const authReducer = (state: AuthState, action: AuthAction): AuthState => {
  switch (action.type) {
    case 'LOGIN':
      return { isAuthenticated: true };
    case 'LOGOUT':
      return { isAuthenticated: false };
    default:
      return state;
  }
}
```

```
const AuthProvider: React.FC<AuthProviderProps> = ({ children }) => {
  const [state, dispatch] = useReducer(authReducer, { isAuthenticated: false });
  const [usuario, setUsuario] = useState({});
  const [token, setToken] = useState({});
  const [rol, setRol] = useState({});
  const [dpi, setDpi] = useState({});

  return (
    <AuthContext.Provider value={{ state, dispatch, usuario, setUsuario, token, setToken, rol, setRol, dpi, setDpi }}>
      {children}
    </AuthContext.Provider>
  );
}
```

3.1.2. AvisoExitoso.tsx / AvisoError.tsx y AvisoErrorInterface.ts /AvisoExitosoInterface.ts

Se utilizo una interface para delimitrar y establecer un tipo sobre los props que se le estarán pasando al componente para mostrar el modal. Es lo mismo tanto para el de “éxito” y “error”, cambian los colores y resultados de operación.

Error:

```
TS AvisoErrorTypes.ts X
proyecto2 > src > AvisoError > TS AvisoErrorTypes.ts > [?] AvisoErrorInterface
1  export type AvisoErrorInterface = {
2    show: boolean;
3    handleClose: () => void;
4    mensaje: string;
5  }
```

```

TS AvisoError.tsx X
proyecto2 > src > AvisoError > TS AvisoError.tsx > ...
1 import React from 'react';
2 import Badge from 'react-bootstrap/Badge';
3 import Stack from 'react-bootstrap/Stack';
4 import { Modal, Button } from 'react-bootstrap';
5 import { AvisoErrorInterface } from './AvisoErrorTypes';
6
7 function AvisoError({ show, handleClose, mensaje }: AvisoErrorInterface) {
8     return (
9         <Modal show={show} onHide={handleClose}>
10             <Modal.Header closeButton>
11                 <Modal.Title>Resultado de la operación</Modal.Title>
12                 <Stack direction="horizontal" gap={2}>
13                     <Badge pill bg="danger">Error</Badge>
14                 </Stack>
15             </Modal.Header>
16             <Modal.Body>
17                 <p>{mensaje}</p>
18             </Modal.Body>
19             <Modal.Footer>
20                 <Button variant="secondary" onClick={handleClose}>
21                     Cerrar
22                 </Button>
23             </Modal.Footer>
24         </Modal>
25     );
26 }
27
28 export default AvisoError;

```

Éxito:

```

TS AvisoExitosoInterface.ts X
proyecto2 > src > AvisoExitoso > TS AvisoExitosoInterface.ts >
1 export interface AvisoExitosoProps {
2     show: boolean;
3     handleClose: () => void;
4     mensaje: string;
5     mensaje2: string;
6 }

```

```

TS AvisoExitoso.tsx X
proyecto2 > src > AvisoExitoso > TS AvisoExitoso.tsx > ...
1 import React from 'react';
2 import Badge from 'react-bootstrap/Badge';
3 import Stack from 'react-bootstrap/Stack';
4 import { Modal, Button } from 'react-bootstrap';
5 import { AvisoExitosoProps } from './AvisoExitosoInterface';
6
7 function AvisoExitoso({ show, handleClose, mensaje, mensaje2 }: AvisoExitosoProps) {
8     return (
9         <Modal show={show} onHide={handleClose}>
10             <Modal.Header closeButton>
11                 <Modal.Title>Resultado de la operación</Modal.Title>
12                 <Stack direction="horizontal" gap={2}>
13                     <Badge pill bg="success">Éxito</Badge>
14                 </Stack>
15             </Modal.Header>
16             <Modal.Body>
17                 <p>{mensaje}</p>
18                 <p>{mensaje2}</p>
19             </Modal.Body>
20             <Modal.Footer>
21                 <Button variant="secondary" onClick={handleClose}>
22                     Cerrar
23                 </Button>
24             </Modal.Footer>
25         </Modal>
26     );
27 }
28
29 export default AvisoExitoso;

```

3.1.3. AvisoLogin.tsx

Se define el tipo **React.FC** para especificar que **AvisoLogin** es un componente de tipo función. Y usarlo como tal.

```
TS AvisoLogin.tsx X
proyecto2 > src > AvisoLogin > TS AvisoLogin.tsx > ...
1  import React from 'react';
2  import Card from 'react-bootstrap/Card';
3  import Button from 'react-bootstrap/Button';
4
5  const AvisoLogin: React.FC = () => {
6    return (
7      <Card className="text-center container mt-5">
8        <Card.Body>
9          <Card.Title>¡Atención!</Card.Title>
10         <Card.Text>
11           Debes iniciar sesión para acceder a esta página.
12         </Card.Text>
13         <Button href="/" variant="primary">Iniciar Sesión</Button>
14       </Card.Body>
15     </Card>
16   );
17 };
18
19 export default AvisoLogin;
```

3.1.4. SelectorCategorías.tsx

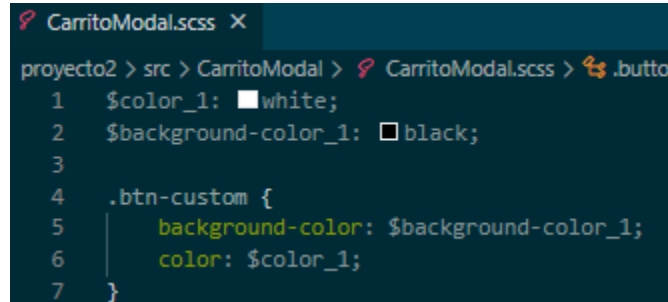
En este caso se crearon los tipos para que el componente los reciba como props y así poder delimitar cuales son los tipos de datos que queremos recibir en el componente.

```
TS SelectorCategorías.tsx X
proyecto2 > src > SelectorCategorías > TS SelectorCategorías.tsx > ...
1  import React from 'react';
2  import Select from 'react-select';
3
4  type CategorySelectProps = {
5    categories: string[];
6    selectedCategory: string;
7    onChange: (selectedCategory: string) => void;
8  };
9
10 function CategorySelect({ categories, selectedCategory, onChange }: CategorySelectProps) {
11   const options = categories.map((category) => ({
12     value: category,
13     label: category,
14   }));
15
16   return (
17     <Select
18       value={{ value: selectedCategory, label: selectedCategory }}
19       options={[
20         { value: 'Todos', label: 'Todos' },
21         ...options,
22       ]}
23       onChange={(selectedOption) => onChange(selectedOption!.value as string)}
24     />
25   );
26 }
27
28 export default CategorySelect;
```

4. Uso de SCSS

4.1. CarritoModal.scss

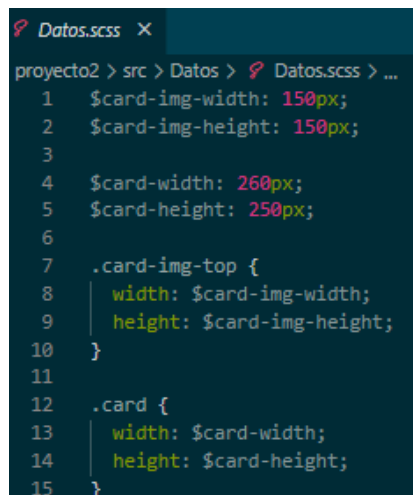
Este archivo se encuentra en ‘./CarritoModal/CarritoModal.scss’, se utilizó para poder modificar los colores tanto de letras como del fondo de un botón. Este es el estilo que se aplicó al botón de comprar en el carrito de compras.

A screenshot of a code editor showing the content of the CarritoModal.scss file. The file is located at 'proyecto2 > src > CarritoModal > CarritoModal.scss'. The code defines two variables, \$color_1 and \$background-color_1, and a class selector .btn-custom that uses these variables to set the background-color and color of a button.

```
proyecto2 > src > CarritoModal > CarritoModal.scss > .button
1  $color_1: white;
2  $background-color_1: black;
3
4  .btn-custom {
5      background-color: $background-color_1;
6      color: $color_1;
7  }
```

4.2. Datos.scss y DatosCompras.scss

Este archivo se encuentra en ‘./Datos/Datos.scss’ y ‘./DatosCompras/DatosCompras.scss’, fue utilizado para determinar los estilos de las cards e imágenes que se mostraban en la lista de productos, específicamente para la lista de productos de un usuario (no administrador). DatosCompras.scss fue usado para mostrar la lista de compras o la bitácora del usuario con su historial.

A screenshot of a code editor showing the content of the Datos.scss file. The file is located at 'proyecto2 > src > Datos > Datos.scss'. The code defines variables for card dimensions and class selectors for .card-img-top and .card, using these variables to set width and height.

```
proyecto2 > src > Datos > Datos.scss > ...
1  $card-img-width: 150px;
2  $card-img-height: 150px;
3
4  $card-width: 260px;
5  $card-height: 250px;
6
7  .card-img-top {
8      width: $card-img-width;
9      height: $card-img-height;
10 }
11
12 .card {
13     width: $card-width;
14     height: $card-height;
15 }
```

4.3. Header.scss

Este archivo se encuentra en ‘./Header / Header.scss’ y se utilizo para realizar el diseño de los botones del header. Estos estilos aplicando tanto para los usuarios y administradores.

```
Header.scss X
proyecto2 > src > Header > Header.scss > .button-spacing
1  $spacing: 10px;
2  $paddingVertical: 5px;
3  $paddingHorizontal: 10px;
4  $lineHeight: 1;
5
6  .button-spacing {
7    margin-right: $spacing;
8    padding: $paddingVertical $paddingHorizontal;
9    line-height: $lineHeight;
10 }
```

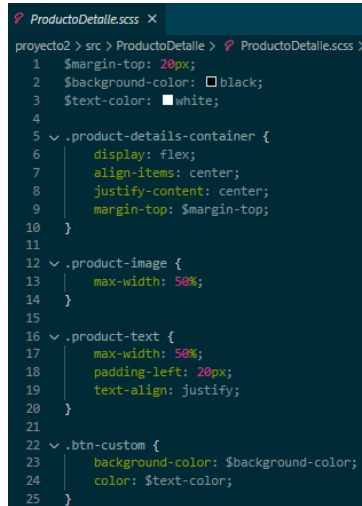
4.4. Perfil.scss

Se encuentra en ‘./Perfil /Perfil.scss’, se aplicaron en el componente que muestra el perfil del usuario con todos sus datos, controla el contenedor y las columnas de este, en el cual se encuentran los diferentes inputs para que el usuario pueda conseguir modificar sus datos.

```
Perfil.scss
proyecto2 > src > Perfil > Perfil.scss > ...
1  $margin-top: 20px;
2  $column-width: 35%;
3  $padding-right: 20px;
4
5  .container {
6    display: flex;
7    align-items: left;
8    justify-content: left;
9    margin-top: $margin-top;
10 }
11
12 .left-column {
13   width: 30%;
14   float: left;
15   padding-right: $padding-right;
16   margin-left: 0;
17 }
18
19 .right-column {
20   width: $column-width;
21   float: left;
22   padding-right: $padding-right;
23 }
24
25 .right-column2 {
26   width: $column-width;
27   float: left;
28   padding-right: $padding-right;
29 }
```


4.5. ProductoDetalle.scss

Este archivo se encuentra en ‘./ProductoDetalle / ProductoDetalle.scss’, se encarga del estilo y de la forma en que se muestra el producto seleccionado con todos sus detalles, este estilo está aplicado únicamente para la página de detalles de productos para los usuarios (no administradores).



```
ProductoDetalle.scss X
proyecto2 > src > ProductoDetalle > ProductoDetalle.scss >
1 $margin-top: 20px;
2 $background-color: black;
3 $text-color: white;
4
5 .product-details-container {
6   display: flex;
7   align-items: center;
8   justify-content: center;
9   margin-top: $margin-top;
10 }
11
12 .product-image {
13   max-width: 50%;
14 }
15
16 .product-text {
17   max-width: 50%;
18   padding-left: 20px;
19   text-align: justify;
20 }
21
22 .btn-custom {
23   background-color: $background-color;
24   color: $text-color;
25 }
```

4.6. ProductoDetalleAdmin.scss y ProductosNuevo.scss

Este archivo se encuentra en ‘./ProductoDetalleAdmin/ProductoDetalleAdmin.scss’ y ‘./ProductosNuevo/ProductosNuevo.scss’, como el archivo anterior, este también se encarga del estilo de ambos componentes y de la forma en que se muestra el producto seleccionado con todos sus detalles, este estilo está aplicado únicamente para la página de detalles de productos para los administradores.

```

ProductoDetalleAdmin.scss X
proyecto2 > src > ProductoDetalleAdmin > ProductoDetalleAdmin.scss
1 $margin-top: 20px;
2 $column-width: 30%;
3 $right-column-width: 35%;
4 $padding-right: 20px;
5 $margin-left: 0;
6
7 .product-details-container {
8     display: flex;
9     align-items: left;
10    justify-content: left;
11    margin-top: $margin-top;
12 }
13
14 .left-column {
15     width: $column-width;
16     float: left;
17     padding-right: $padding-right;
18     margin-left: $margin-left;
19 }
20
21 .right-column {
22     width: $right-column-width;
23     float: left;
24     padding-right: $padding-right;
25 }
26
27 .right-column2 {
28     width: $right-column-width;
29     float: left;
30 }

```

4.7. Registro.scss

Este archivo se encuentra en ‘./Registro/Registro.scss’, se encarga de aplicar los estilos para el formulario de ‘Registarse’ en el que maneja varios inputs y formularios para que el cliente ingrese cualquier de sus datos personales.

```

Registro.scss X
proyecto2 > src > Registro > Registro.scss > product-details-container
1 $margin-top: 20px;
2 $column-width: 30%;
3 $right-column-width: 35%;
4 $padding-right: 20px;
5 $margin-left: 0;
6
7 $product-details-margin: 20px;
8 $product-details-width: 30%;
9 $product-details-padding-right: 20px;
10 $product-details-margin-left: 0;
11
12 .no-spinners::-webkit-inner-spin-button,
13 .no-spinners::-webkit-outer-spin-button {
14     -webkit-appearance: none;
15     margin: 0;
16 }
17
18 .product-details-container {
19     display: flex;
20     align-items: left;
21     justify-content: left;
22     margin-top: $product-details-margin;
23 }
24
25 .left-column {
26     width: $product-details-width;
27     float: left;
28     padding-right: $product-details-padding-right;
29     margin-left: $product-details-margin-left;
30 }
31
32 .right-column {
33     width: $product-details-width;
34     float: left;
35     padding-right: $product-details-padding-right;
36 }
37
38 .right-column2 {
39     width: $product-details-width;
40     float: left;
41 }
42
43 }

```

5. Enlaces de los proyectos

5.1. API's en Render.com

<https://proyecto1-kelvin-and-lester.onrender.com/>

5.2. Proyecto React en Netlify.com

<https://iridescent-maamoul-1acb03.netlify.app/>