

**“AÑO DEL BICENTENARIO DEL PERÚ: 200 AÑOS DE INDEPENDENCIA”.**



**UNIVERSIDAD NACIONAL DE SAN AGUSTÍN DE  
AREQUIPA**

**ESCUELA PROFESIONAL DE CIENCIA DE LA  
COMPUTACIÓN  
COMPUTACIÓN EN LA NUBE**

---

## **Informe del Proyecto Final**

---

***Alumno:***

Chuctaya Elme, Milagros  
Herrera Cooper, Miguel Alexander  
Huisa Flores, César Gabriel  
Maguiña del Castillo, Brayan Jean Pool

***Docente:***

Mg. Iquiria Becerra, Diego Alonso

1 de Febrero del 2021



# Índice

<b>1. Configuración del enrutador React</b>	<b>3</b>
<b>2. Crear la barra de navegación</b>	<b>6</b>
<b>3. Creación de la página de inicio de la APP</b>	<b>10</b>
<b>4. Configuración de React Context API</b>	<b>15</b>
<b>5. Adición de la Función de Carrito de Compras</b>	<b>17</b>
<b>6. Configuración de Firebase</b>	<b>20</b>
<b>7. Autenticación - Login</b>	<b>26</b>
7.1. Creación del componente Login.js . . . . .	26
7.2. Edición de Header.js . . . . .	28
7.3. Evidencias del Login . . . . .	30
<b>8. Carrito de Compras</b>	<b>31</b>
8.1. Archivo CartItem . . . . .	31
8.2. Archivo CartTotal.js . . . . .	32
8.3. Integración y Demostración . . . . .	33
<b>9. Creación ChatBot</b>	<b>34</b>
9.1. Importar SimpleForm en App.js . . . . .	34
9.2. Instalación de React Simple Chatbot . . . . .	34
9.3. Import react-simple-chatbot en SimpleForm.js . . . . .	35
9.4. Agregar pasos de usuario al componente ChatBot . . . . .	35
9.5. Agregar propiedades de configuración a React Simple Chatbot . . . . .	36
<b>10. Cronograma de actividades</b>	<b>38</b>
<b>11. Deploy</b>	<b>39</b>
11.1. Github Pages . . . . .	39
11.2. Deploy de la aplicación . . . . .	39
<b>12. Referencias</b>	<b>41</b>

## Índice de figuras

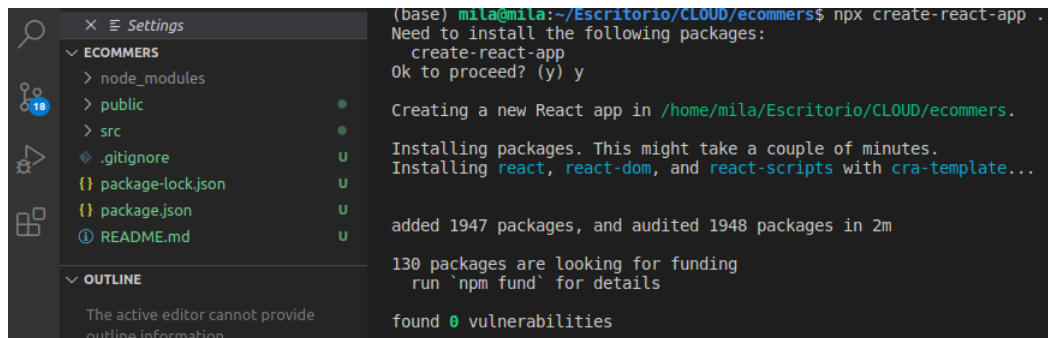
1.	Ejecucion del comando de creacion de React. . . . .	3
2.	Datos del localhost de la aplicacion creada. . . . .	3
3.	Pantalla de React. . . . .	4
4.	De esta manera se procedera a seguir configurando. . . . .	4
5.	Resultado del paquete react-router-dom. . . . .	4
6.	Resultado con React Router. . . . .	5
7.	Ejemplo de Interfaz de la barra de búsqueda . . . . .	9
8.	Ejemplo de Interfaz de pantalla de inicio . . . . .	10
9.	Ejemplo de Interfaz con el componente Product . . . . .	14
10.	Botón de Login . . . . .	30
11.	Login . . . . .	30
12.	Carrito de Compras . . . . .	33
13.	Aplicación despleada en la nube . . . . .	40

## 1. Configuración del enrutador React

Primeramente se va a instalar y configurar la aplicación React, para ello simplemente escribiremos lo siguiente en la terminal:

```
npx create-react-app .
```

Hasta que aparezca en la terminal "Happy Hacking".



```
(base) mila@mila:~/Escritorio/CLOUD/ecommerce$ npx create-react-app .
Need to install the following packages:
  create-react-app
Ok to proceed? (y) y

Creating a new React app in /home/mila/Escritorio/CLOUD/ecommerce.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template...

added 1947 packages, and audited 1948 packages in 2m

130 packages are looking for funding
  run `npm fund` for details

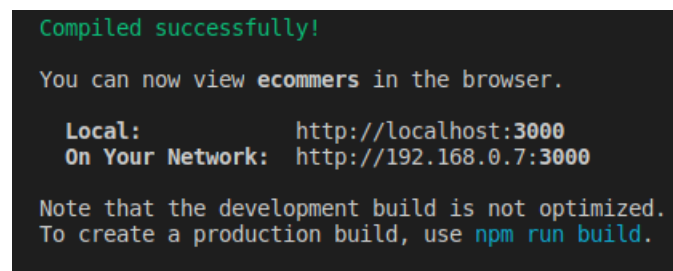
found 0 vulnerabilities
```

Figura 1: Ejecucion del comando de creacion de React.

Ahora que tenemos nuestra aplicación React instalada, podemos iniciarla. En la terminal, escribiremos el siguiente comando. Este comando iniciará la aplicación React.

```
npm start
```

Cargara la ejecución y luego si se hizo todo correctamente, saldrá la siguiente pantalla en la ventana en nuestro navegador.



```
Compiled successfully!

You can now view ecommerce in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://192.168.0.7:3000

Note that the development build is not optimized.
To create a production build, use npm run build.
```

Figura 2: Datos del localhost de la aplicacion creada.

Seguidamente se mostrara la pantalla de bienvenida de React. Esta sera nuestra plantilla para poder construir el Ecommerce, por lo que se necesitara hacer unos cambios para que poder comenzar.

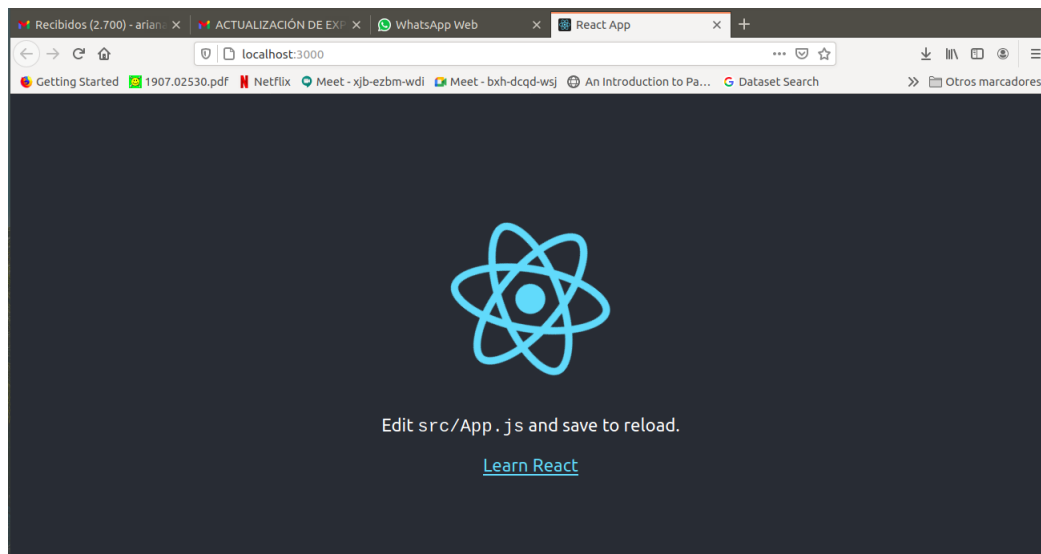


Figura 3: Pantalla de React.

Realizando los cambios necesarios quedo de la siguiente manera.

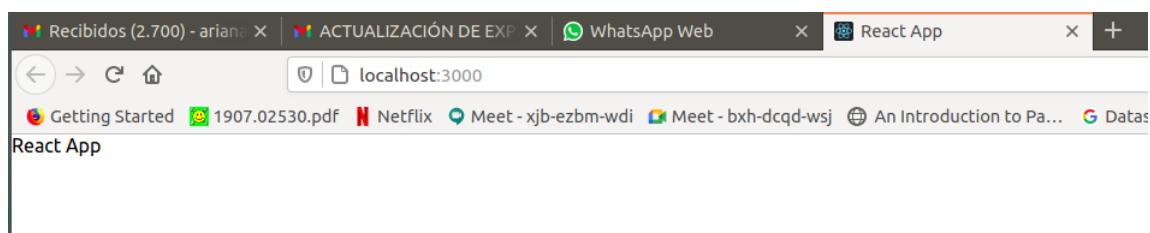


Figura 4: De esta manera se procedera a seguir configurando.

Algo muy importante a considerar en una aplicación React es la navegación (pasar de una página a otra) de los usuarios. Dado que React es una aplicación de una sola página, no admite varias rutas de forma predeterminada.

Para ello hay un paquete llamado **react-router-dom** que nos permite crear rutas para nuestro proyecto React. La configuración es una vez y, luego, cada vez que agrega una nueva página, solo necesita informar al enrutador. Se ingresara el siguiente comando para instalar react-router-dom

```
npm install react-router-dom
```

Cuando se complete la instalación, aparecera el siguiente resultado:

```
(base) mila@mila:~/Escritorio/CLOUD/ecommerce$ npm install react-router-dom
up to date, audited 1992 packages in 15s

130 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
(base) mila@mila:~/Escritorio/CLOUD/ecommerce$
```

Figura 5: Resultado del paquete react-router-dom.

Se creara un componente, simplemente cree un nuevo archivo. Lo cual se llamara Home.js y también un archivo Home.css para el estilo del componente.

Ahora, agregaremos un codigo basico en el archivo Home.js ahora:

```
1 import React from "react";
2 function Home() {
3   return <div className="home">Hello</div>;
4 }
5 export default Home;
```

Configurar el React Router. En primer lugar, importaremos las dependencias, usando el siguiente código en la parte superior de App.js:

```
1 import { BrowserRouter as Router, Route, Switch } from "react-router-dom";
```

Una vez terminado de importar el código, se podra usar React Router en el archivo. Ahora se configurara React Router se hara unos cambios con código en App.js, el cual quedara asi:

```
1 import React from "react";
2 import "./App.css";
3 import { BrowserRouter as Router, Route, Switch } from "react-router-dom";
4 import Home from "./Home";
5 function App() {
6   return (
7     <div className="app">
8       <Router>
9         <Switch>
10          <Route path="/">
11            <Home />
12          </Route>
13        </Switch>
14      </Router>
15    </div>
16  );
17 }
18 export default App;
```

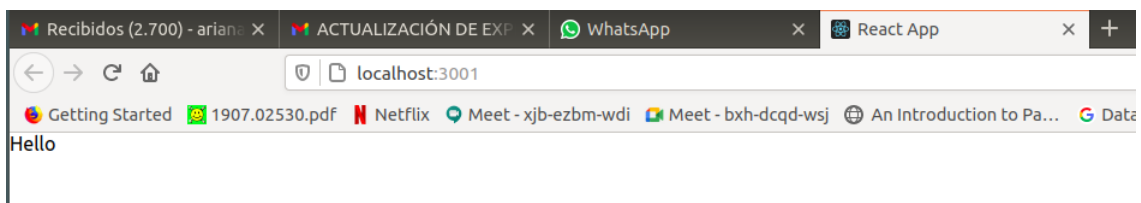


Figura 6: Resultado con React Router.

En toda esta primera parte se enfoco en encajar toda la aplicación en el componente del enrutador, de modo que cada componente sea parte del enrutador y tenga acceso al enrutador.

## 2. Crear la barra de navegación

Para esto, utilizaremos un paquete de iconos y necesitamos *Material Icons* para utilizarlos. Para ello utilizaremos el siguiente comando.

```
npm install @material-ui/core @material-ui/icons
```

Ahora se puede usar para mostrar los iconos SVG que proporciona Material UI ( es una biblioteca de UI muy popular para React) la cuál tiene muchos componentes prediseñados como íconos.

Luego creemos un nuevo componente llamado **Header.js** y creemos un nuevo archivo y lo llamaremos **Header.css** tal como hicimos con el componente de inicio. En cada componente seguiremos los mismos pasos.

Lo que sigue es que tenemos que inicializar la placa de caldera del componente usando “rfce” y seguir la convención BEM e incluir el archivo CSS y actualizar los nombres de las clases. Antes de comenzar a diseñarlo teneos que incluirlo en el enrutador para que podamos mostrarlo. En **App.js** donde pusimos la Ruta para la ruta “/”, incluyamos también el componente Encabezado.

```
1
2 <Route path="/">
3   <Header />
4   <Home />
5 </Route>
```

Regresamos al **Header.js** y configuramos el diseño de nuestra barra de búsqueda.

```
1
2 import React from "react";
3 import "./Header.css";
4 import SearchIcon from "@material-ui/icons/Search";
5 import ShoppingBasketIcon from "@material-ui/icons/ShoppingBasket";
6
7 function Header() {
8   return (
9     <div className="header">
10
11       
15
16
17       <div className="header__search">
18         <input className="header__searchInput" type="text" />
19         <SearchIcon className="header__searchIcon" />
20       </div>
21
22       <div className="header__nav">
23
24         <div className="header__option">
25           <span className="header__optionLineOne">Hello Guest</span>
26           <span className="header__optionLineTwo">Sign In</span>
27         </div>
```

```

28
29
30
31     <div className="header__option">
32         <span className="header__optionLineOne">Returns</span>
33         <span className="header__optionLineTwo">& Orders</span>
34     </div>
35
36
37
38     <div className="header__option">
39         <span className="header__optionLineOne">Your</span>
40         <span className="header__optionLineTwo">Prime</span>
41     </div>
42
43
44     <div className="header__optionBasket">
45         <ShoppingBasketIcon />
46         <span className="header__optionLineTwo header__basketCount">
47             0
48         </span>
49     </div>
50
51 </div>
52 </div>
53 );
54 }
55
56 export default Header;

```

Y también tenemos el archivo **Header.css** para que la parte de estilo esté lista.

```

1
2 .header {
3     height: 60px;
4     display: flex;
5     align-items: center;
6     background-color: #131921;
7     position: sticky;
8     top: 0;
9     z-index: 100;
10 }
11
12 .header__logo {
13     width: 100px;
14     object-fit: contain;
15     margin: 0 20px;
16     margin-top: 18px;
17 }
18
19 .header__search {
20     display: flex;
21     flex: 1;
22     align-items: center;
23     border-radius: 24px;
24 }
25
26 .header__searchInput {

```



```
27 height: 12px;
28 padding: 10px;
29 border: none;
30 width: 100%;
31 }
32
33 .header__searchIcon {
34   padding: 5px;
35   height: 22px !important;
36   background-color: #cd9042;
37 }
38
39 .header__optionLineOne {
40   font-size: 10px;
41 }
42
43 .header__optionLineTwo {
44   font-size: 13px;
45   font-weight: 800;
46 }
47
48 .header__optionBasket {
49   display: flex;
50   align-items: center;
51   color: white;
52 }
53
54 .header__basketCount {
55   margin-left: 10px;
56   margin-right: 10px;
57 }
58
59 .header__nav {
60   display: flex;
61   justify-content: space-evenly;
62 }
63
64 .header__option {
65   display: flex;
66   flex-direction: column;
67   margin-left: 10px;
68   margin-right: 10px;
69   color: white;
70 }
```

En resumen lo que se hizo fue:

- Una barra de navegación. En el primer bloque de código presentamos el diseño. Es como HTML básico, pero en React lo llamamos JSX.
- Configuramos que queremos el logotipo a la izquierda, la barra de búsqueda en el centro y las otras opciones a la derecha, que luego hacemos que suceda con nuestros archivos CSS.
- También hemos utilizado íconos de Material UI en la barra de navegación, como el ícono de búsqueda y los íconos de la cesta, que fueron importados del paquete Material UI Icons.

- En CSS configuramos la propiedad de visualización de la barra de navegación para que se flexione de modo que los elementos se alineen uno al lado del otro, cambiamos los colores de fondo, ajustamos el cuadro de búsqueda y luego diseñamos las opciones.

Finalmente con el código obtenemos este resultado:



Figura 7: Ejemplo de Interfaz de la barra de búsqueda

### 3. Creación de la página de inicio de la APP

Se creara el comopnente Home, el cual, servirá como raiz de la aplicación.

Primero se crea el componente en el router de react, para lo cual se creará el archivo llamado **Home.js**

```
1 import React from "react";
2 import "../Home.css";
3
4 function Home() {
5   return (
6     <div className="home">
7       <div className="home__container">
8         
12      </div>
13    </div>
14  );
15 };
16
17 export default Home;
```

Con eso se crea el componente home que servirá como componente principal en la aplicación y se añadirá su respectivo archivo de estilo llamado **Home.css**

```
1 .home {
2   display: flex;
3   justify-content: center;
4   margin-left: auto;
5   margin-right: auto;
6   max-width: 1500px;
7 }
8 .home__image {
9   width: 100%;
10  z-index: -1;
11  margin-bottom: -150px;
12  mask-image: linear-gradient(to bottom, rgba(0, 0, 0, 1), rgba(0, 0, 0, 0));
13 }
```

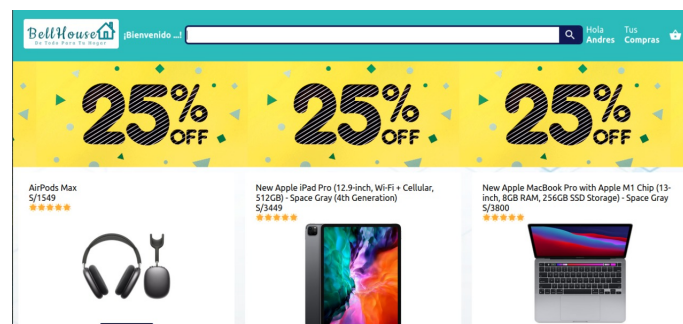


Figura 8: Ejemplo de Interfaz de pantalla de inicio

Ahora se debe crear un componente productos el cual será contenido por la pantalla de inicio y permitirá tratar a los productos como componentes individuales para lo cual se crea el archivo **Product.js**

```

1 import React from "react";
2 import "../Product.css";
3
4 function Product({ id, title, image, price, rating }) {
5
6   return (
7     <div className="product">
8       <div className="product__info">
9         <p>{title}</p>
10        <p className="product__price">
11          <small>${</small>
12          <strong>{price}</strong>
13        </p>
14        <div className="product__rating">
15          {Array(rating)
16            .fill()
17            .map((_, i) => (
18              <p><,</p>
19            ))}
20        </div>
21      </div>
22
23      <img src={image} alt="" />
24
25      <button>Add to Basket</button>
26    </div>
27  );
28 }
29
30 export default Product;
31

```

Con su respectivo css:

```

1 .product {
2   display: flex;
3   flex-direction: column;
4   align-items: center;
5   justify-content: flex-end;
6   margin: 10px;
7   padding: 20px;
8   width: 100%;
9   max-height: 400px;
10  min-width: 100px;
11  background-color: white;
12  z-index: 1;
13 }
14
15 .product__rating {
16   display: flex;
17 }
18
19 .product > img {
20   max-height: 200px;

```

```

21 width: 100%;
22 object-fit: contain;
23 margin-bottom: 15px;
24 }
25
26 .product > button {
27   background: #f0c14b;
28   border: 1px solid;
29   margin-top: 10px;
30   border-color: #a88734 #9c7e31 #846a29;
31   color: #111;
32 }
33
34 .product__price {
35   margin-top: 5px;
36 }
37
38 .product__info {
39   height: 100px;
40   margin-bottom: 15px;
41 }
42

```

Todo esto basando la distribución y la relación de aspecto con otras tiendas online.

A el componente Product se le pasan cierta información para que pueda renderizar cada cuadro de manera independiente, esta información incluye:

- ID de producto
- Nombre de producto
- Precio
- Imagen
- Calificación

Ahora se añaden manualmente algunos componentes Product para tener una vista de como funcionaria cuando la app funcione en totalidad

```

1 import React from "react";
2 import "./Home.css";
3 import Product from "./Product";
4
5 function Home() {
6   return (
7     <div className="home">
8       <div className="home__container">
9         

```

```

15     <div className="home__row">
16         <Product
17             id="12321341"
18             title="The Lean Startup: How Constant Innovation Creates
Radically Successful Businesses Paperback"
19             price={11.96}
20             rating={5}
21             image="https://images-na.ssl-images-amazon.com/images/I/51
Zymoq7UnL._SX325_BO1,204,203,200_.jpg"
22         />
23         <Product
24             id="49538094"
25             title="Kenwood kMix Stand Mixer for Baking, Stylish Kitchen
Mixer with K-beater, Dough Hook and Whisk, 5 Litre Glass Bowl"
26             price={239.0}
27             rating={4}
28             image="https://images-na.ssl-images-amazon.com/images/I/810%2
BGNdkzKL._AC_SX450_.jpg"
29         />
30     </div>
31
32     <div className="home__row">
33         <Product
34             id="4903850"
35             title="Samsung LC49RG90SSUXEN 49' Curved LED Gaming Monitor"
36             price={199.99}
37             rating={3}
38             image="https://images-na.ssl-images-amazon.com/images/I/71
Swqqe7XAL._AC_SX466_.jpg"
39         />
40         <Product
41             id="23445930"
42             title="Amazon Echo (3rd generation) | Smart speaker with Alexa,
Charcoal Fabric"
43             price={98.99}
44             rating={5}
45             image="https://media.very.co.uk/i/very/
P6LTG_SQ1_0000000071_CHARCOAL_SLf?$300x400_retinamobilex2$"
46         />
47         <Product
48             id="3254354345"
49             title="New Apple iPad Pro (12.9-inch, Wi-Fi, 128GB) - Silver (4
th Generation)"
50             price={598.99}
51             rating={4}
52             image="https://images-na.ssl-images-amazon.com/images/I/816
ctt5WV5L._AC_SX385_.jpg"
53         />
54     </div>
55
56     <div className="home__row">
57         <Product
58             id="90829332"
59             title="Samsung LC49RG90SSUXEN 49' Curved LED Gaming Monitor -
Super Ultra Wide Dual WQHD 5120 x 1440"
60             price={1094.98}
61             rating={4}

```

```

62     image="https://images-na.ssl-images-amazon.com/images/I/6125
    mFrzr6L._AC_SX355_.jpg"
63     />
64   </div>
65 </div>
66 </div>
67 );
68 }
69
70 export default Home;
71

```

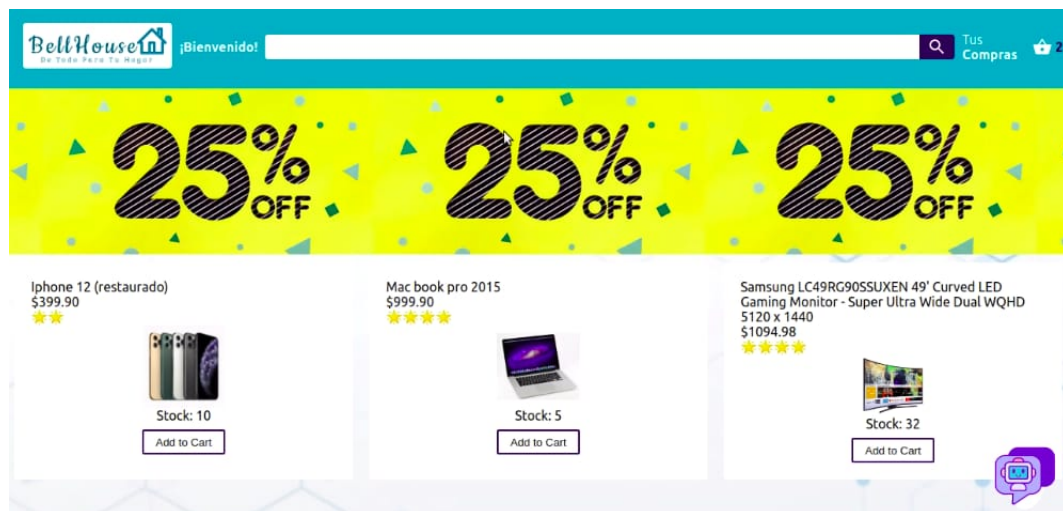


Figura 9: Ejemplo de Interfaz con el componente Product

Ahora se añade una corrección en el estilo principal para lo cual se accede a **index.css** y se añade correcciones en fuente y espaciado:

```

1 body {
2   background-color: rgb(234, 237, 237);
3   margin: 0;
4   font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", "Roboto", "
    Oxygen",
5     "Ubuntu", "Cantarell", "Fira Sans", "Droid Sans", "Helvetica Neue",
6     sans-serif;
7   -webkit-font-smoothing: antialiased;
8   -moz-osx-font-smoothing: grayscale;
9 }
10
11 code {
12   font-family: source-code-pro, Menlo, Monaco, Consolas, "Courier New",
13     monospace;
14 }

```

Y con eso ya se tendría listo el componente raíz **Home** para implementar la app

## 4. Configuración de React Context API

La API de contexto es una parte muy importante de React. Nos ayuda a hacer estados a nivel de aplicación y podemos obtener los datos de esos estados a través de cualquier componente, sin embargo existen muchas alternativas, una de ellas es Redux; aunque los expertos sugieren iniciar con React Context API.

Creemos un archivo llamado **StateProvider.js** y escribimos lo siguiente :

```
1 import React, { createContext, useContext, useReducer } from "react";
2
3 // Prepara la capa de datos
4 export const StateContext = createContext();
5
6 // Envuelv nuestra aplicacion y proporcione la capa de datos
7 export const StateProvider = ({ reducer, initialState, children }) => (
8   <StateContext.Provider value={useReducer(reducer, initialState)}>
9     {children}
10   </StateContext.Provider>
11 );
12
13 // Extrae informacion de la capa de datos
14 export const useStateValue = () => useContext(StateContext);
```

La configuración de React Context API es más una repetición y prácticamente lo mismo en todos los proyectos.

Ahora creamos un archivo **reducer.js**, aquí es donde se define todos los estados de la aplicación y las acciones para realizar cambios en el estado.

```
1 export const initialState = {
2   basket: [],
3   user: null
4 };
5
6 // Selector
7 export const getBasketTotal = (basket) =>
8   basket?.reduce((amount, item) => item.price + amount, 0);
9
10 const reducer = (state, action) => {
11   console.log(action);
12   switch (action.type) {
13     default:
14       return state;
15   }
16 };
17
18 export default reducer;
```



Veamos qué sucede en este fragmento de código.

- El reductor se declaran todos los estados del nivel de la aplicación y los que puede utilizar posteriormente.
- En **initialState**, declaramos los estados que vamos a utilizar. En este caso, basket y users.
- La función reductora real toma un estado y una acción para realizar operaciones en el estado.
- Exportamos el **reductor**.

Es necesario usar este reductor para administrar el estado de nuestro nivel de aplicación. Para hacerlo, introducimos todo en index.js.

```
1 import React from "react";
2 import ReactDOM from "react-dom";
3 import "./index.css";
4 import App from "./App";
5 import * as serviceWorker from "./serviceWorker";
6 import reducer, { initialState } from "./reducer";
7 import { StateProvider } from "./StateProvider";
8
9 ReactDOM.render(
10   <React.StrictMode>
11     <StateProvider initialState={initialState} reducer={reducer}>
12       <App />
13     </StateProvider>
14   </React.StrictMode>,
15   document.getElementById("root")
16 );
17 serviceWorker.unregister();
```

- Importamos pocas cosas: **reducer** e **initialState** de **reducer.js** y luego importamos el componente **StateProvider** de **StateProvider.js**.
- Incluimos el componente **App** con **StateProvider** para que los niños también tengan acceso a los estados.
- Pasamos el **reducer** y **initialState** al **StateProvider**.
- Ahora **State** sabe cuál es el estado inicial y qué reductor también.

## 5. Adición de la Función de Carrito de Compras

Básicamente, se necesita importar el *estado*, cuando surge la necesidad de enviar eventos.

Para agregar la funcionalidad de carrito de compra, es necesario realizar cambios en el archivo **Product.js**

Primeramente, importamos el estado y el envío usando el nexo **useStateValue ()**.

```
import useStateValue from "../stateProvider";
```

Seguidamente se crea una función **Product** que se encarga de obtener la colección de datos y mostrarla en la interfaz.

```
1 import React from 'react'
2 import { db } from '../firebase'
3 import './Product.css'
4
5 function Product(props) {
6
7     const addToCart = () => {
8         const cartItem = db.collection('cart-items').doc(props.id);
9         cartItem.get().then((doc) => {
10             if (doc.exists) {
11                 cartItem.update({
12                     quantity: parseInt(doc.data().quantity) + 1
13                 })
14             } else {
15                 cartItem.set({
16                     title: props.title,
17                     image: props.image,
18                     price: props.price,
19                     quantity: 1
20                 })
21             }
22         })
23     })
24
25 }
26 return (
27     <div className="Product">
28         <div className="Product-description">
29             <span className="Product-title">{props.title}</span>
30             <span className="Product-price">${props.price}</span>
31             <span className="Product-rating">
32                 <p>    </p><p>    </p><p>    </p><p>    </p><p>    </p>
33             </span>
34         </div>
35         <img src={props.image} />
36         <button onClick={addToCart} >Aadir al carro</button>
37     </div>
38 )
39 }
40
41 export default Product
```

Lo que sucede en el anterior fragmento de código es lo siguiente :

- Se agrega un tipo de acción al caso del interruptor, lo llamamos **Agregar al Carrito**.
- Configuramos el estado, actualizamos la canasta, pero aquí usamos el operador de spread (...) para que no se pierda el estado anterior. Y así es como agregamos elementos a la matriz de la canasta en nuestro estado en reductor.

Luego, se actualizará la barra de navegación para mostrar el número de elementos presentes actualmente en el carrito. Entonces se debe modificar el archivo **Header.js**

Reemplazamos el 0 codificado en la parte del carrito del componente Encabezado , y ahora el código quedaría de la siguiente manera :

```

1 import React from 'react'
2 import './Header.css'
3 import Logo from './logo2.png';
4 import SearchIcon from '@material-ui/icons/Search';
5 import ShoppingBasketIcon from '@material-ui/icons/ShoppingBasket';
6 import {
7   BrowserRouter as Router,
8   Switch,
9   Route,
10  Link
11 } from "react-router-dom";
12
13 function Header(props) {
14
15   const getCount = () => {
16     // init count with 0
17     // loop through all the items
18     // add quantity to count
19     // return it
20     let count = 0;
21     props.cartItems.forEach(item => {
22       count += parseInt(item.cartItem.quantity)
23     });
24     return count;
25   }
26
27   return (
28     <div className="Header">
29       { /* Logo */ }
30       <Link to="/">
31         <div className="Header-logo">
32           <img src={Logo} />
33         </div>
34       </Link>
35       { /* Address */ }
36       <div className="Header-optionAddress">
37         { /* icon */ }
38         <div className="Header-option">
39
40           <span className="Header-optionLineTwo"> Bienvenido
41         ...!</span>
42         </div>

```

```

43     </div>
44     { /* Search */ }
45     <div className="Header-search">
46         <input className="Header-searchInput" type="text" />
47         <div className="Header-searchIconContainer">
48             <SearchIcon />
49         </div>
50     </div>
51     <div className="Header-navItems">
52         { /* Login name */ }
53         <div className="Header-option">
54             <span className="Header-optionLineOne">Hola</span>
55             <span className="Header-optionLineTwo">Andres</span>
56         </div>
57         { /* Orders */ }
58         <div className="Header-option">
59             <span className="Header-optionLineOne">Tus</span>
60             <span className="Header-optionLineTwo">Compras</span>
61         </div>
62         { /* Cart */ }
63         <Link to="/cart">
64             <div className="Header-optionCart">
65                 <ShoppingBasketIcon />
66                 <span className="Header-cartCount">{getCount()}</span>
67             </div>
68         </Link>
69     </div>
70
71
72 </div>
73 )
74 }
75
76 export default Header

```

Entonces se introdujo lo siguiente :

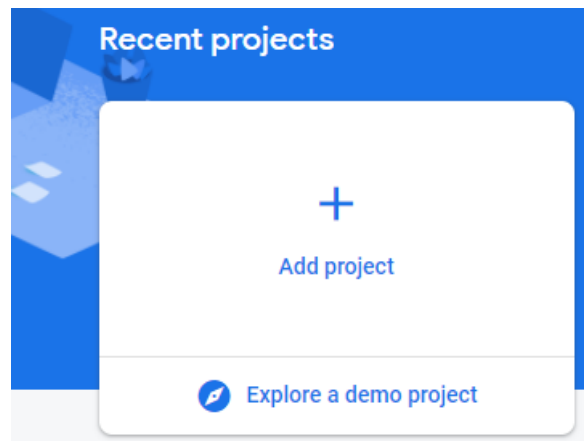
- Se ha incorporado los estados y las funciones de envío para que podamos enviar acciones al reductor.
- Eliminamos el 0 codificado de la canasta y se agrega la longitud del carrito, que dará la cantidad de artículos en la matriz del carrito de compra.
- Configuramos los enlaces para el pago. Esto se debe a que los necesitaremos en la siguiente sección. El componente de enlace viene con react-router-dom para que podamos enlazar a rutas sin recargar páginas.



## 6. Configuración de Firebase

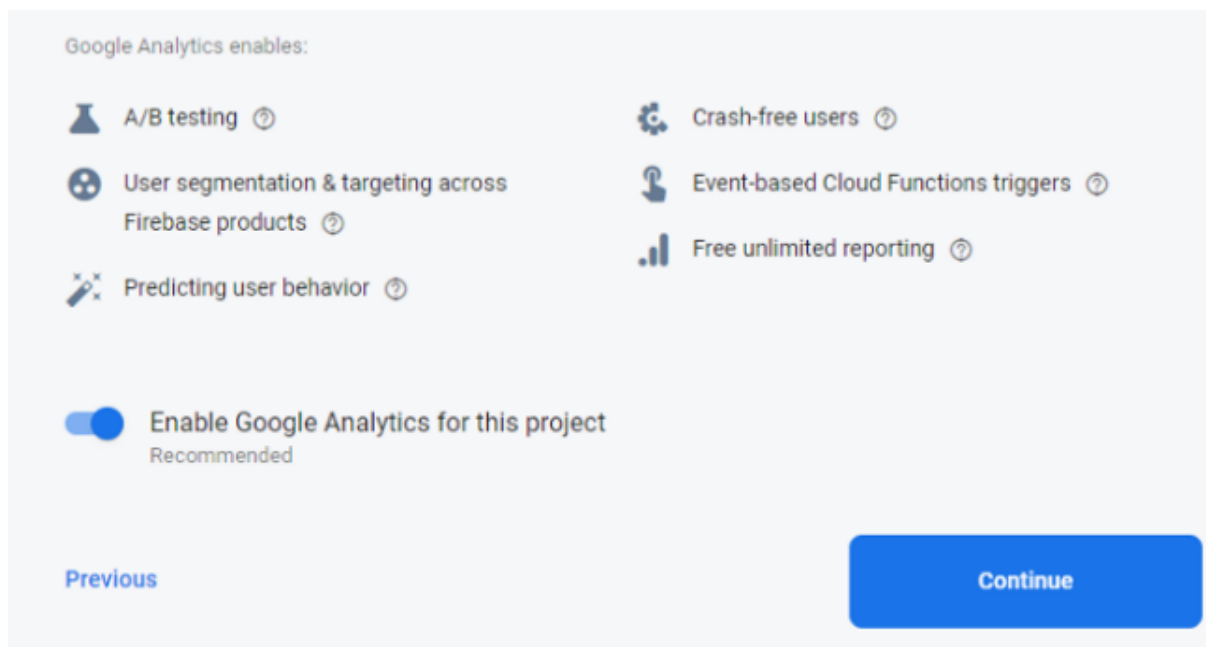
Una vez dentro de la consola de Firebase, seguimos los siguientes pasos.

1. Haga clic en "Agregar proyecto"

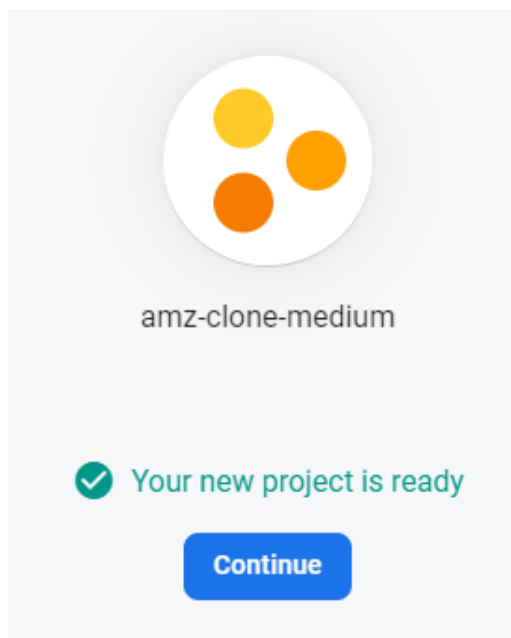


2. Se establece un nombre al proyecto.

3. Habilitar Google Analytics para el proyecto no es necesario ya que no hará ninguna diferencia. Pero la imagen a continuación es un ejemplo del proceso.

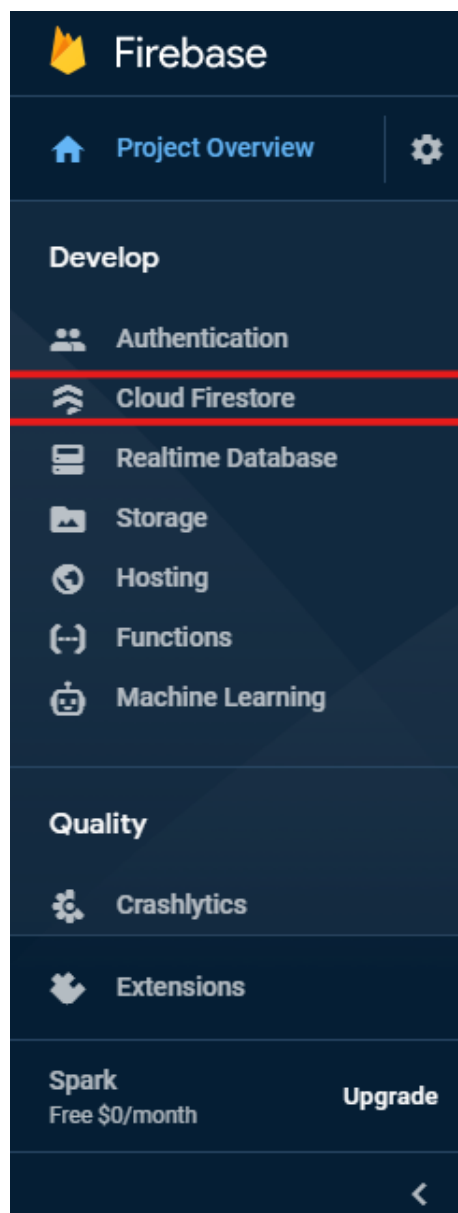


4. Una vez que vea la imagen de abajo, se crea su proyecto de Firebase y se debe hacer clic en continuar.



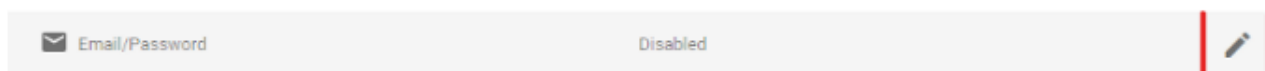
5. Ahora configuremos este proyecto de acuerdo con nuestras necesidades.

- En el panel derecho, seleccionar Cloud Firestore Cloud, la cual es la base de datos de Firebase.
- Hacer clic en Crear base de datos .
- Seleccionar Iniciar en modo de prueba .
- Hacer clic en Habilitar y esperar a que Google suministre Cloud Firestore.

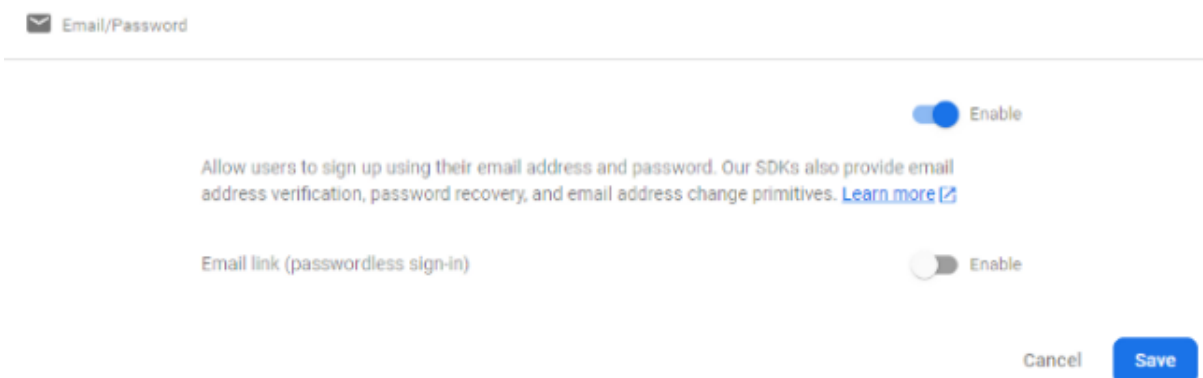


## 6. Ahora configuremos la autenticación

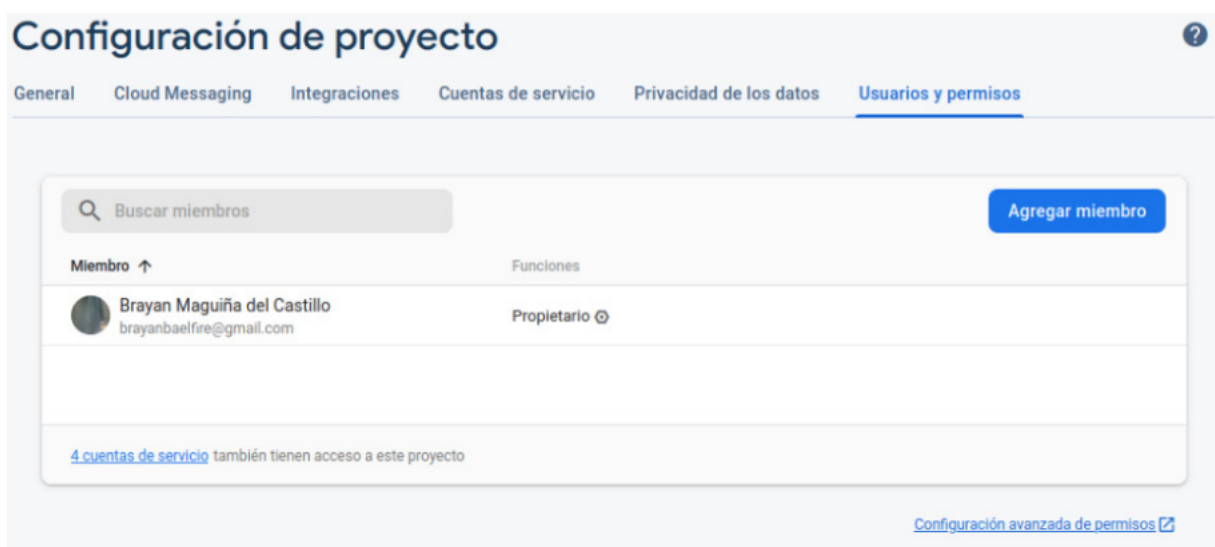
- Seleccionar Autenticación en el panel lateral.
- Seleccionar Configurar método de inicio de sesión.
- Hacer clic en el ícono de lápiz cerca de Correo electrónico / Contraseña.



## 7. Habilitar y hacer clic en Guardar .



## 8. Ejemplo de usuario

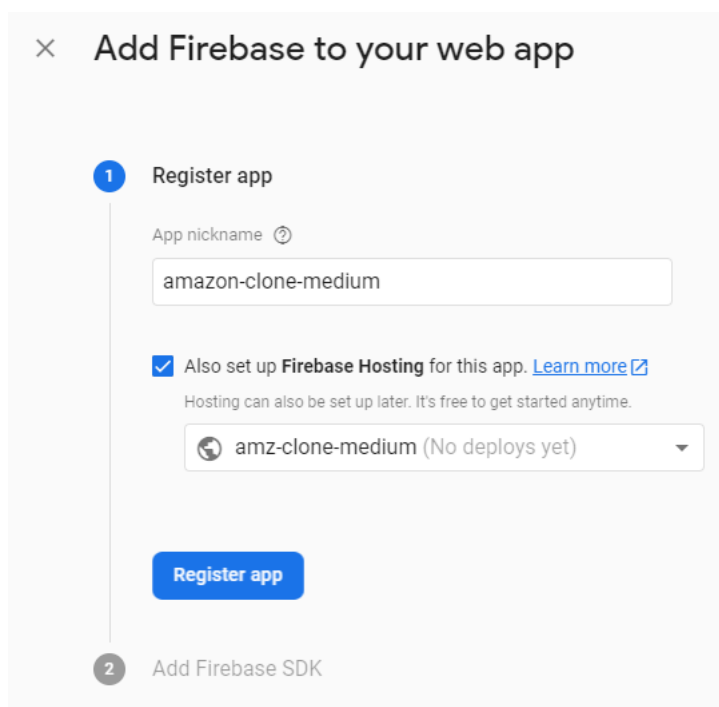




9. Ahora hacer clic en el ícono de Configuración en el panel lateral y seleccionar Configuración del proyecto para poder configurar el proyecto Firebase con nuestro proyecto React.
10. Hacer clic en el tercer icono (Web) en la parte inferior



11. Ingresar el nombre del proyecto y seleccionar también configurar Firebase Hosting .La siguiente imagen muestra un ejemplo.



12. Presionar **Siguiente** hasta que vuelva a su proyecto de **Firebase**.
13. En la aplicación React se debe instalar dependencias llamadas **firebase** y **firebase-tools**.

```
npm install -g firebase-tools npm install firebase firebase login
```

14. En Firebase y seleccionemos la opción de configuración.

#### Firebase SDK snippet

☐ Automático ⓘ ☐ CDN ⓘ ☒ Configuración ⓘ

Copia y pega estas secuencias de comandos en la parte inferior de la etiqueta <body> antes de usar cualquier servicio de Firebase:

```
// For Firebase JS SDK v7.20.0 and later, measurementId is optional
const firebaseConfig = {
  apiKey: "AIzaSyDrn_HVeS8BmFcG5EG1Wqo9bjDc0UKYzN8",
  authDomain: "ecomers-4ab27.firebaseio.com",
  projectId: "ecomers-4ab27",
  storageBucket: "ecomers-4ab27.appspot.com",
  messagingSenderId: "822990952674",
  appId: "1:822990952674:web:2d3b821ee83cb6381726bb",
  measurementId: "G-GYZCTEK342"
};
```

15. Copiar la configuración completa y luego pegarlo en el archivo llamado firebase.js .
16. Hacer uso del paquete firebase que se instalo a través de npm y editar el archivo firebase.js para que podamos usar la autenticación y la base de datos del proyecto.

```
1 import firebase from 'firebase'
2
3 // For Firebase JS SDK v7.20.0 and later, measurementId is optional
4 const firebaseApp = firebase.initializeApp( {
5   apiKey: "AIzaSyDrn_HVeS8BmFcG5EG1Wqo9bjDc0UKYzN8",
6   authDomain: "ecomers-4ab27.firebaseio.com",
7   projectId: "ecomers-4ab27",
8   storageBucket: "ecomers-4ab27.appspot.com",
9   messagingSenderId: "822990952674",
10  appId: "1:822990952674:web:2d3b821ee83cb6381726bb",
11  measurementId: "G-GYZCTEK342"
12 })
13
14 const db = firebase.firestore();
15
16 export { db }
```

## 7. Autenticación - Login

Se avanzó el login y por razones de tiempo no se pudo incluir a la interfaz principal.

### 7.1. Creación del componente Login.js

Creamos un nuevo componente, llamado **Inicio de sesión**, para ello se debe de crear dos archivos nuevos, **Login.js** y **Login.css**

```

1 import React, { useState } from 'react';
2 import './Login.css'
3 import { Link, useHistory } from "react-router-dom";
4 import { auth } from "./firebase";
5
6 function Login() {
7
8     const [email, setEmail] = useState('');
9     const [password, setPassword] = useState('');
10
11     return (
12         <div className='login'>
13             <Link to='/'>
14                 <img
15                     className="login__logo"
16                     src='Cooper/Escritorio/5to/Cloud_Computing/Prc_12/
Bellhouse_logo.png'
17                 />
18             </Link>
19
20             <div className='login__container'>
21                 <h1>Sign-in</h1>
22
23                 <form>
24                     <h5>E-mail</h5>
25                     <input type='text' value={email} onChange={e =>
setEmail(e.target.value)} />
26
27                     <h5>Contraseña</h5>
28                     <input type='password' value={password} onChange={e =>
setPassword(e.target.value)} />
29
30                     <button type='submit' className='login__signInButton'>
Iniciar Sesión</button>
31                 </form>
32
33                 <p>
34                     Al iniciar sesión, acepta las condiciones de uso y
venta de BellHouse. Por favor
35                     consulte nuestro Aviso de privacidad, nuestro Aviso de
cookies y nuestro Aviso de anuncios basados en intereses.
36                 </p>
37
38                 <button className='login__registerButton'>Crear cuenta
BellHouse</button>
39             </div>
40         </div>

```

```

41 )
42 }
43
44 export default Login

```

Se agrega el estilo con **Login.css**.

```

1 .login {
2   background-color: white;
3   height: 100vh;
4   display: flex;
5   flex-direction: column;
6   align-items: center;
7 }
8
9 .login__logo {
10  margin-top: 20px;
11  margin-bottom: 20px;
12  object-fit: contain;
13  width: 100px;
14  margin-right: auto;
15  margin-left: auto;
16 }
17
18 .login__container {
19  width: 300px;
20  height: fit-content;
21  display: flex;
22  flex-direction: column;
23  border-radius: 5px;
24  border: 1px solid lightgray;
25  padding: 20px;
26 }
27
28 .login__container > h1 {
29  font-weight: 500;
30  margin-bottom: 20px;
31 }
32
33 .login__container > form > h5 {
34  margin-bottom: 5px;
35 }
36
37 .login__container > form > input {
38  height: 30px;
39  margin-bottom: 10px;
40  background-color: white;
41  width: 98%;
42 }
43
44 .login__container > p {
45  margin-top: 15px;
46  font-size: 12px;
47 }
48
49 .login__signInButton {
50  background: #f0c14b;
51  border-radius: 2px;

```

```

52     width: 100%;
53     height: 30px;
54     border: 1px solid;
55     margin-top: 10px;
56     border-color: #a88734 #9c7e31 #846a29;
57   }
58
59   .login__registerButton {
60     border-radius: 2px;
61     width: 100%;
62     height: 30px;
63     border: 1px solid;
64     margin-top: 10px;
65     border-color: darkgray;

```

Entonces el funcionamiento del componente de **Inicio de Sesión** se realiza de la siguiente manera:

- Tenemos dos estados locales que realizan un seguimiento de los valores de los cuadros de texto.
- Siempre que cambia el valor de los cuadros de texto, se cambia el valor de estado.

Se debe agregarlo a React Router y actualizar la barra de navegación.

En a **App.js**se ingresa esta ruta antes de la ruta principal `/`.

```

1 <Route path="/login">
2   <Login />
3 </Route>

```

## 7.2. Edición de Header.js

Ahora nos dirigimos a **Header.js** y usamos el contenido.

```

1 import React from "react";
2 import "./Header.css";
3 import SearchIcon from "@material-ui/icons/Search";
4 import ShoppingBasketIcon from "@material-ui/icons/ShoppingBasket";
5 import { Link } from "react-router-dom";
6 import { useStateValue } from "./stateProvider";
7 import { auth } from "./firebase";
8
9 function Header() {
10   const [{ basket, user }, dispatch] = useStateValue();
11
12   const handleAuthentication = () => {
13     if (user) {
14       auth.signOut();
15     }
16   }
17
18   return (
19     <div className="header">
20       <Link to="/">
21         <img
22           className="header__logo"

```

```

23     src='Cooper/Escritorio/5to/Cloud_Computing/Prc_12/Bellhouse_logo.
    png'
24     />
25     </Link>
26
27     <div className="header__search">
28         <input className="header__searchInput" type="text" />
29         <SearchIcon className="header__searchIcon" />
30     </div>
31
32     <div className="header__nav">
33         <Link to={!user && '/login'}>
34             <div onClick={handleAuthenticaton} className="header__option">
35                 <span className="header__optionLineOne">Hola {!user ? 'Guest' :
user.email}</span>
36                 <span className="header__optionLineTwo">{user ? 'Sign Out' : '
Sign In'}</span>
37             </div>
38         </Link>
39
40         <Link to='/orders'>
41             <div className="header__option">
42                 <span className="header__optionLineOne">Retornar</span>
43                 <span className="header__optionLineTwo">& Pedidos</span>
44             </div>
45         </Link>
46
47
48         <div className="header__option">
49             <span className="header__optionLineOne">Tu</span>
50             <span className="header__optionLineTwo">Principal</span>
51         </div>
52
53         <Link to="/checkout">
54             <div className="header__optionBasket">
55                 <ShoppingBasketIcon />
56                 <span className="header__optionLineTwo header__basketCount">
57                     {basket?.length}
58                 </span>
59             </div>
60         </Link>
61     </div>
62 </div>
63 );
64 }
65
66 export default Header;

```

Las cosas adicionales son :

- Se comprueba si el usuario está autenticado. En caso afirmativo, se saluda al usuario; de lo contrario, el usuario tiene la opción de iniciar sesión.
- La función **handleAuthentication** comprueba si el usuario ha iniciado sesión o no, en caso contrario, cierra la sesión cuando el usuario hace clic en el saludo.

### 7.3. Evidencias del Login

A continuación se muestra la interfaz resultante de la autenticación.

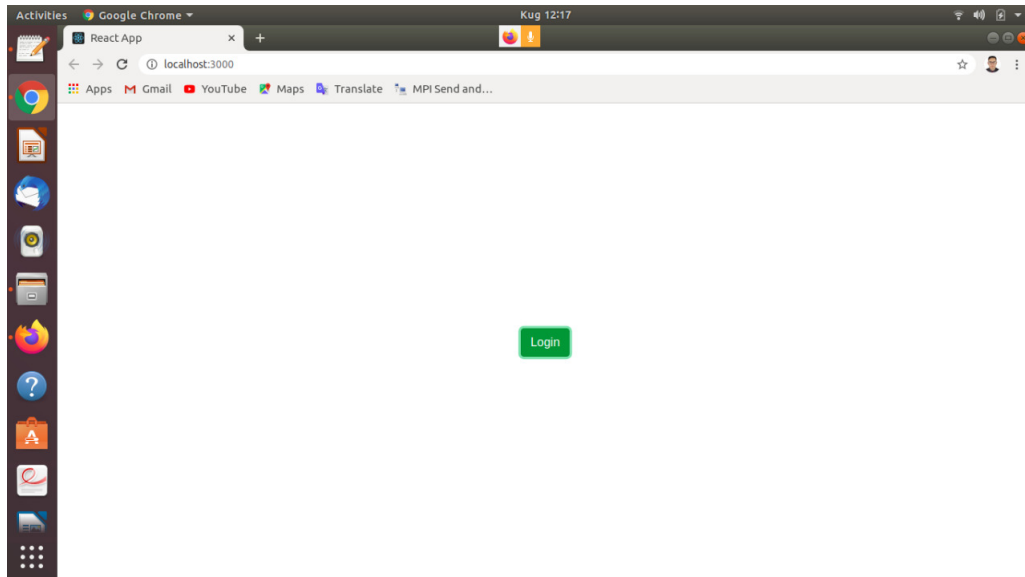


Figura 10: Botón de Login

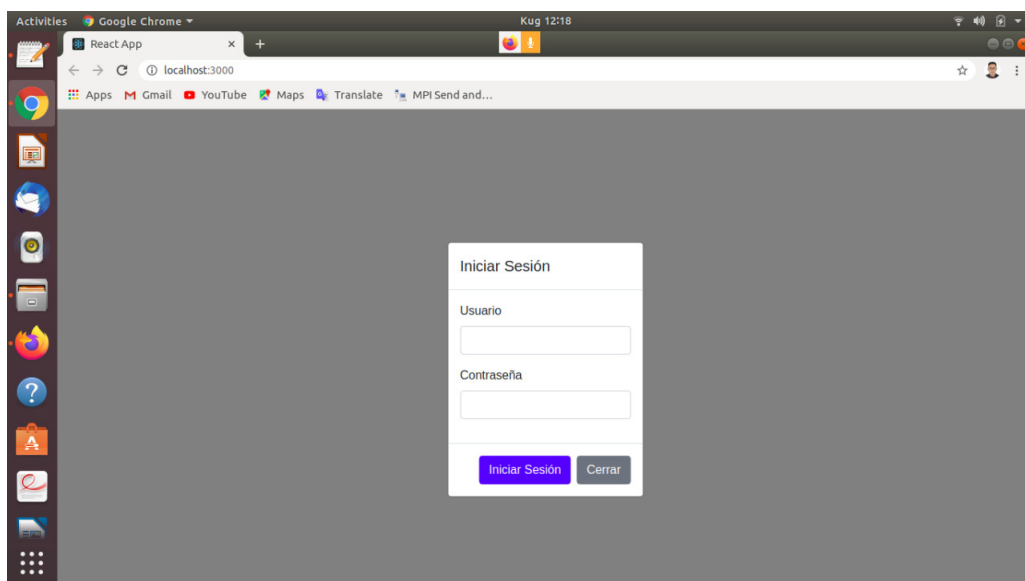


Figura 11: Login

## 8. Carrito de Compras

Dado que la mayor parte del proyecto está configurado, hacer la canasta o carrito de compra resulta ser muy fácil ya que se realiza y trabaja con múltiples componentes.

### 8.1. Archivo CartItem

En este archivo mostramos los productos de la base de datos en Firebase, con lo que cual usuario podra seleccionar o eliminar su compra

```

1 import React from 'react'
2 import './CartItem.css'
3 import { db } from './firebase';
4
5 function CartItem({ id, cartItem }) {
6
7     const deleteItem = () => {
8         db.collection('cart-items').doc(id).delete();
9     }
10
11
12     const changeQuantity = (newQuantity) => {
13         db.collection('cart-items').doc(id).update({
14             quantity: parseInt(newQuantity)
15         })
16     }
17
18
19     let options = [];
20
21     for(let i = 1; i < Math.max(cartItem.quantity+1, 20); i++){
22         options.push(<option value={i}>Cantidad: {i}</option>)
23     }
24
25     return (
26         <div className="CartItem">
27             <div className="CartItem-image">
28                 <img src={cartItem.image} />
29             </div>
30             <div className="CartItem-info">
31                 <div className="CartItem-title">
32                     {cartItem.title}
33                 </div>
34                 <div className="CartItem-actions">
35                     <div className="CartItem-quantity">
36                         <select
37                             onChange={ (e) => changeQuantity(e.target.value)
38
39                             value={cartItem.quantity}>
40                             {options}
41                         </select>
42                     </div>
43                     <div
44                         onClick={deleteItem}
45                         className="CartItem-delete">
46                         Eliminar

```



```

46         </div>
47     </div>
48
49     </div>
50     <div className="CartItem-price">
51         ${cartItem.price}
52     </div>
53 </div>
54 )
55 }
56
57 export default CartItem

```

## 8.2. Archivo CartTotal.js

En este archivo creamos una función **CartTotal** que permite contabilizar el total de productos seleccionados por el usuario para realizar la compra, luego suma los montos por cada productos y retorna el precio final a pagar.

```

1 import React from 'react'
2 import './CartTotal.css'
3
4 function CartTotal(props) {
5
6
7     const getCount = () => {
8         // init count with 0
9         // loop through all the items
10        // add quantity to count
11        // return it
12        let count = 0;
13        props.cartItems.forEach(item => {
14            count += parseInt(item.cartItem.quantity)
15        });
16        return count;
17    }
18
19    const getTotalPrice = () => {
20        let total = 0;
21        props.cartItems.forEach((item)=>{
22            total += (item.cartItem.price * item.cartItem.quantity)
23        })
24        return total;
25    }
26
27    return (
28        <div className="CartTotal">
29            <h3>Total({getCount()} items): s/{getTotalPrice()} </h3>
30            <button>Pagar</button>
31        </div>
32    )
33 }
34
35 export default

```

### 8.3. Integración y Demostración

Para poder integrar las funcionalidades implementadas anteriormente creamos un archivo **Cart.js** donde importamos los archivos **CartItem.js** y **CartTotal.js**

```
1 import React from 'react'
2 import CartItems from './CartItem'
3 import CartTotal from './CartTotal'
4 import './Cart.css'
5
6 function Cart(props) {
7   return (
8     <div className="Cart">
9       <CartItems cartItems={props.cartItems} />
10      <CartTotal cartItems={props.cartItems} />
11    </div>
12  )
13 }
14
15 export default
```

Seguidamente se muestran los resultados de la interfaz

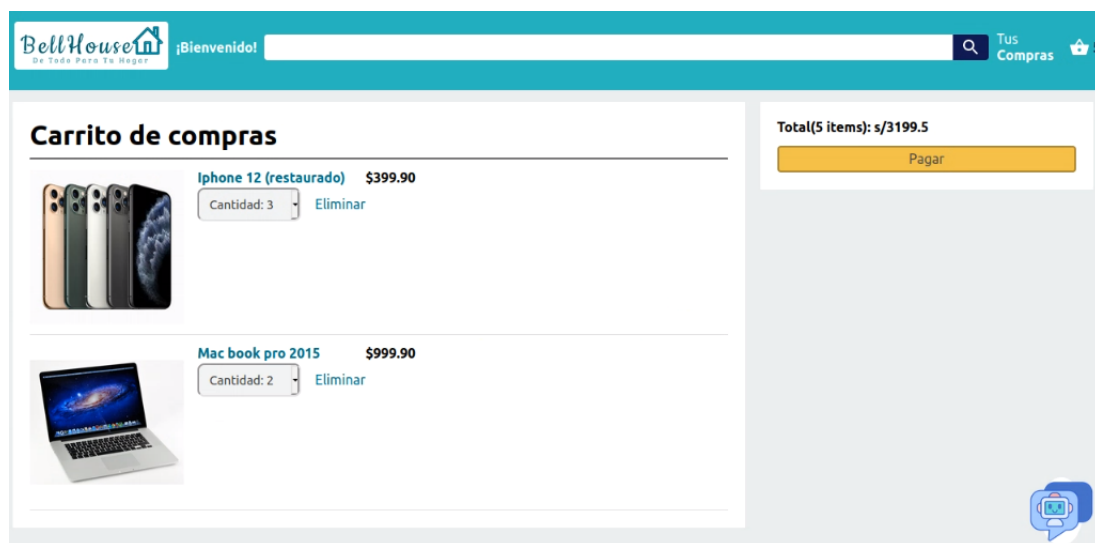


Figura 12: Carrito de Compras

## 9. Creación ChatBot

Para la creación del ChatBot, lo tomamos ya que aproximadamente el 80 % de las empresas requieren chatbots, como parte de una transición más amplia para automatizar los procesos y sistemas comerciales que respaldan el servicio al cliente en la industria del comercio electrónico, los chatbots permiten a los minoristas fomentar mejores experiencias de compra y agilizar una vasta matriz de interacciones complejas.

Para ello se procedió a crear un Chatbot, como una implementación adicional además que se puede aplicar lo que se hizo en clases, se proseguirá a explicar como se creó el Chatbot en la aplicación de Ecommerce.

1. Se creará una nueva carpeta llamada componentes en el directorio src.
2. A continuación, se creará una carpeta llamada chatbot en el directorio de componentes. Todo el código de React chatbot va en components>directorio de chatbot.
3. Luego se creará un archivo llamado SimpleForm.js en el directorio del chatbot para el componente React chatbot.
4. La estructura del archivo es src>components>chatbot>SimpleForm.js.

Ahora se proseguirá a construir el código de creación del ChatBot:

### 9.1. Importar SimpleForm en App.js

```
1 import React, { Component } from 'react';
2 import SimpleForm from "../components/chatbot/SimpleForm";
3
4 class App extends Component {
5   render() {
6     return (
7       <div>
8         <SimpleForm />
9       </div>
10     );
11   }
12 }
13
14 export default App;
```

### 9.2. Instalación de React Simple Chatbot

React Simple Chatbot es un componente de chatbot que se utiliza para crear chats de conversación, es una forma rápida de obtener un chatbot personalizado y compatible con dispositivos móviles en cualquier sitio web. Se instala con el siguiente comando:

```
npm install react-simple-chatbot --save
```

Luego se instala styled-components, además de styled-components, nos permitirá cambiar la temática del chatbot. Lo instalaremos con el siguiente comando:

```
npm install styled-components
```

### 9.3. Import react-simple-chatbot en SimpleForm.js

```

1 import React, { Component } from 'react';
2 import ChatBot from 'react-simple-chatbot';
3
4 class SimpleForm extends Component {
5   render() {
6     return (
7       <ChatBot />
8     );
9   }
10 }
11
12 export default SimpleForm;

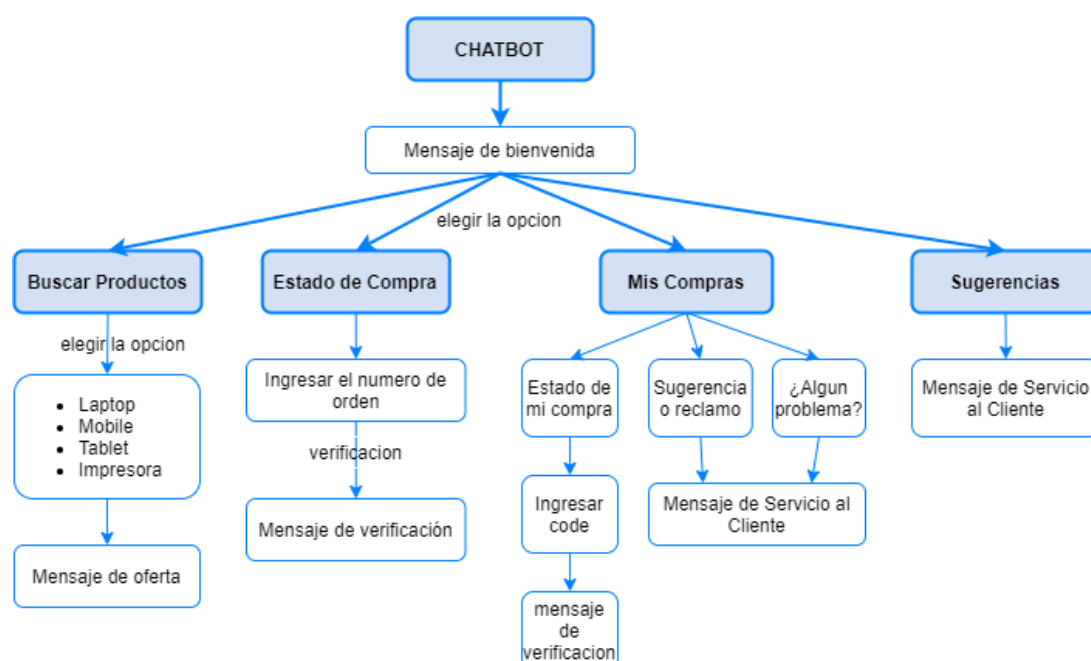
```

### 9.4. Agregar pasos de usuario al componente ChatBot

Se comenzara con los pasos del texto. Estos son los pasos que el Chatbot le indica al usuario. Atributos comunes de los pasos de texto:

- **id (cadena o número):** requerido para cualquier paso.
- **message (cadena o función):** un mensaje para el usuario.
- **trigger (cadena, número o función):** el ID del siguiente paso activado.
- **validator (función):** una función que se utiliza para validar el texto del usuario escrito.
- **end (booleano):** si es verdadero, este paso es el final de la conversación.

El esquema de las opciones que presenta el Chatbot se personalizo de la siguiente manera:



```

1 import React, { Component, useState } from "react";
2 import ChatBot from "react-simple-chatbot";
3 import notificacion from "./chatbot2.png";
4
5 const SimpleForm = () => {
6   const [showChatBot, setShowChatBot] = useState(false);
7   return (
8     {showChatBot && (
9       <ChatBot
10        steps=[
11          {
12            id: "bienvenida",
13            message:
14              "Hola ! Soy Bell, el asistente virtual de BellHouse Estoy
15              para ayudarte!",
16            trigger: "1",
17          },
18          {
19            id: "1",
20            message: "Elige una opcion",
21            options: [
22              { value: 1, label: "Buscar productos", trigger: "product"
23            },
24              { value: 2, label: "Estado de compra", trigger: "estado" },
25              { value: 3, label: "Mis compras", trigger: "compras" },
26              { value: 4, label: "Sugerencias", trigger: "sugerencia" },
27            ],
28            trigger: "2",
29          },
30        ],
31      >
32    );
33   };
34   export default SimpleForm;

```

## 9.5. Agregar propiedades de configuración a React Simple Chatbot

En esta sección se presenta el código de configuración del botón, se logra la activación el botón del chatbot,

```

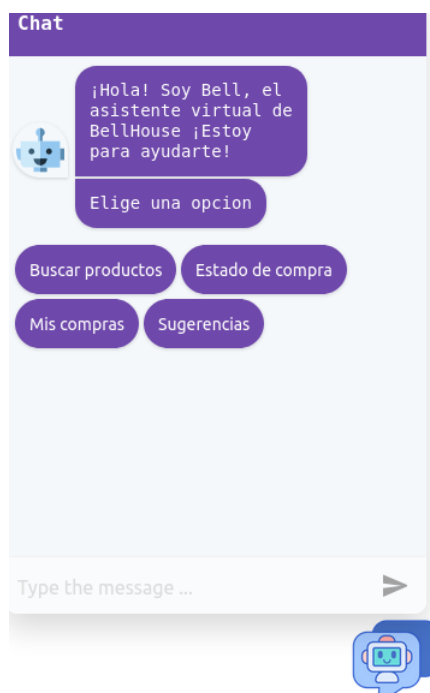
1 const SimpleForm = () => {
2   const [showChatBot, setShowChatBot] = useState(false);
3   return (
4     <div
5       style={{
6         margin: 10,
7         top: "auto",
8         right: 20,
9         bottom: 20,
10        left: "auto",
11        position: "fixed",
12        floating: true,
13        color: "#FFFFFF",
14      }}
15     >
16   );
17 };
18 export default SimpleForm;

```

Por ultimo se dan configuración de estilo para centrar el icono, color o imagen, formato del botón, en el mismo archivo **SimpleForm.js**

```
1 import notificacion from "../chatbot2.png";
2 <div
3   style={{
4     width: "100%",
5     display: "flex",
6     alignItems: "flex-end",
7     flexDirection: "column",
8   }}
9   >
10  <div
11    onClick={() => setShowChatBot(!showChatBot)}
12    style={{
13      borderRadius: "50%",
14      minHeight: "50px",
15      minWidth: "50px",
16      maxWidth: "50px",
17      maxHeight: "50px",
18      backgroundColor: "#ffffff",
19      display: "flex",
20      alignItems: "center",
21      justifyContent: "center",
22    }}
23    >
24    <img
25      style={{ width: "80px", height: "80px" }}
26      src={notificacion}
27    ></img>
28  </div>
29 </div>
```

Finalmente se presenta algunas imágenes como referencia de la implementación del Chatbot.

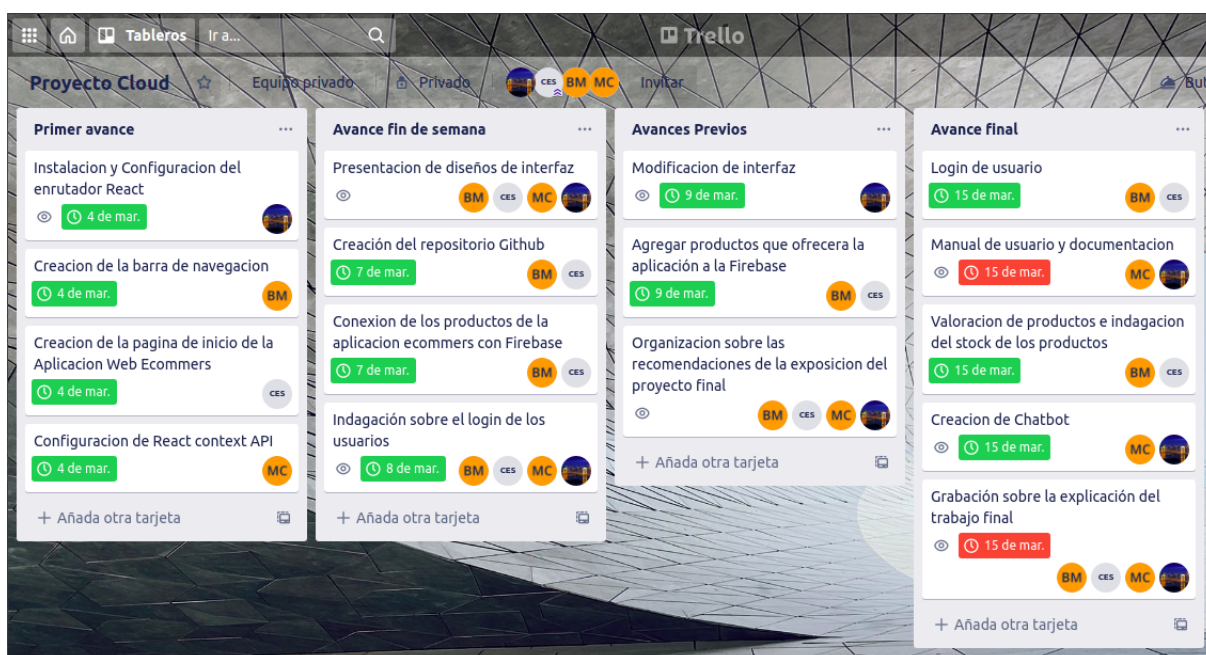


## 10. Cronograma de actividades

Mediante esta plataforma de Trello, se distribuyeron las actividades necesarias para organizar las tareas sobre el desarrollo de esta aplicación web de ecommerce.

En la siguiente imagen se muestra la distribución por secciones, las cuales fueron 4:

- La primera sección trata sobre las primeras tareas de configuración y creación de la aplicación web ecommerce.
- La segunda sección trata sobre un avance en segunda etapa, la cual se avanzó el fin de semana para presentarlo el día martes, se avanzó la parte de interfaz, conexión con Firebase en cuanto a los productos y usuarios.
- La tercera sección trata sobre las recomendaciones para mejorar el proyecto final.
- Por último, la cuarta sección fue la organización de los detalles como login de usuario, stock de los productos, creación de chatbot y sobre la documentación respectiva.



## 11. Deploy

### 11.1. Github Pages

Las GitHub Pages son páginas web públicas alojadas y publicadas fácilmente a través de GitHub mediante su paquete de gh-pages.

### 11.2. Deploy de la aplicación

Para hacer el despliegue en la plataforma de github sólo se necesita:

1. Instalar gh-pages: Para lo cual usaremos el comando:

```
1 npm install gh-pages --save-dev
```

2. Añadir directrices al archivo *package.json*

```
1 {
2   /...
3   "homepage": "http://(nombre de usuario)-github.com.io/(Nombre del
4   repositorio"
5   /...
6   "scripts": {
7     /...
8     "predeploy": "npm run build"
9     "deploy": "gh-pages -d build"
10    /...
11  }
```

3. Ejecutar el deploy: Para lo cual se usa el siguiente comando:

```
1 npm run deploy
```

Una vez hecho esto ya tendremos nuestra app desplegada en la nube de github



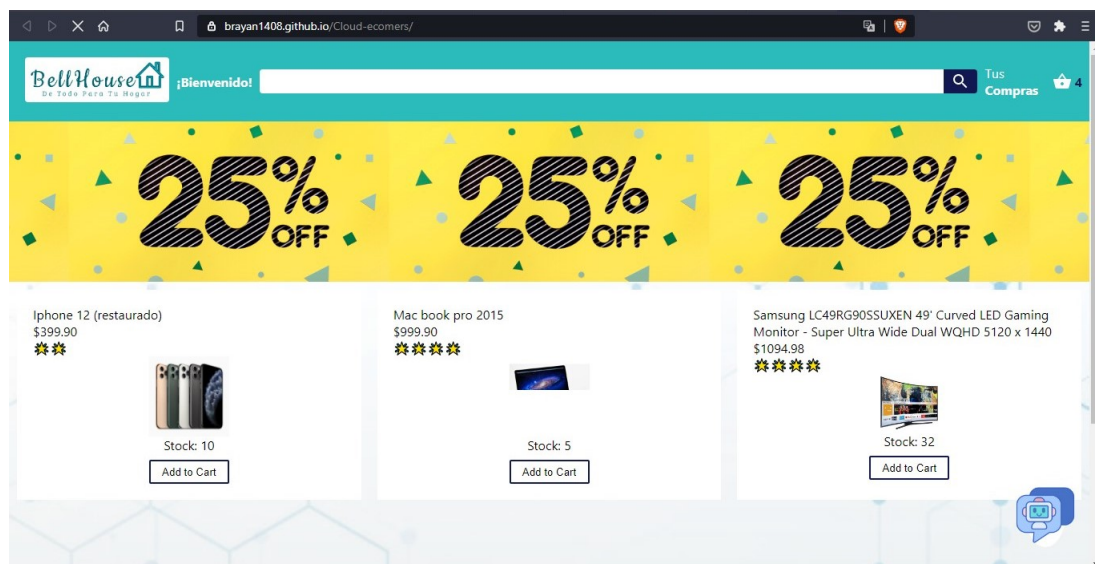


Figura 13: Aplicación desplegada en la nube

## 12. Referencias

1. Link de la grabacion : hacer click
2. Link del repositorio : hacer click
3. Link del Proyecto en la nube