

HÉROES APP

Iniciando proyecto

Creando app:

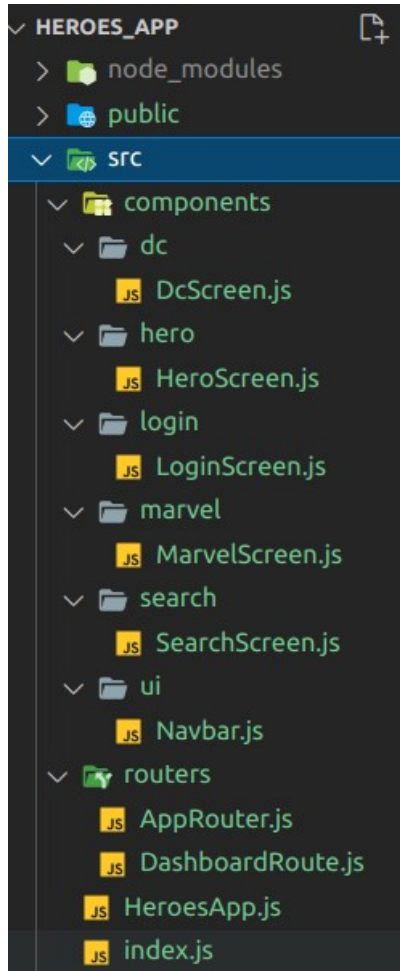
```
$$> npx create-react-app heroes_app
```

instalando paquete para el Router:

```
$$> npm install react-router-dom@6
```

Colocar CDN de estilos de bootstrap

Crear las carpetas:



Navbar:

```
import { Link, NavLink } from "react-router-dom";
```

```
export const Navbar = () => {
  const handleLogout = () => {
    console.log("logout");
  };

```

```
  return (
    <nav className="navbar navbar-expand-sm navbar-dark bg-dark">
      <Link className="navbar-brand" to="/">
        Asociaciones
      </Link>

```

```
      <div className="navbar-collapse">
        <div className="navbar-nav">
          <NavLink className="nav-item nav-link" to="/marvel">
            Marvel
          </NavLink>

```

```
          <NavLink className="nav-item nav-link" to="/dc">
            DC
          </NavLink>
        </div>
      </div>

```

```
    <div className="navbar-collapse collapse w-100 order-3 dual-collapse2 d-flrx justify-content-end">

```

```

<ul className="navbar-nav ml-auto">
  <span className="nav-item nav-link text-info">Y. Daniel</span>
  <button className="nav-item nav-link btn" onClick={handleLogout}>
Logout
  </button>
</ul>
</div>
</nav>
);
};

```

DashboardRoute:

```

import { Routes, Route } from "react-router-dom";
import DcScreen from "../components/dc/DcScreen";

import HeroScreen from "../components/hero/HeroScreen";
import MarvelScreen from "../components/marvel/MarvelScreen";
import SearchScreen from "../components/search/SearchScreen";
import { Navbar } from "../components/ui/Navbar";

const DashboardRoute = () => {
  return (
    <>
      <Navbar />
      <Routes>
        <Route path="/" element={<MarvelScreen />} exact={true} />
        <Route path="marvel" element={<MarvelScreen />} />
        <Route path="dc" element={<DcScreen />} />
        <Route path="hero" element={<HeroScreen />} />
        <Route path="search" element={<SearchScreen />} />
      </Routes>
    </>
  );
};

export default DashboardRoute;

```

AppRouter:

```

import { BrowserRouter, Routes, Route } from "react-router-dom";
import LoginScreen from "../components/login/LoginScreen";
import DashboardRoute from "../DashboardRoute";

const AppRouter = () => {
  return (
    <BrowserRouter>
      <Routes>
        {/* Login sin el Navbar */}
        <Route path="/login" element={<LoginScreen />} />
        <Route path="/*" element={<DashboardRoute />} />
      </Routes>
    </BrowserRouter>
  );
};

export default AppRouter;

```

Navigate push replace – useNavigate

En LoginScreen:

```
import { useNavigate } from "react-router-dom";
```

```
const LoginScreen = () => {
```

```
// Custom hook para la navegación
```

```
const navigate = useNavigate();
```

```
const handleLogin = () => {
```

```
// replace: true => Para no regresar al login
```

```
navigate("/", {
```

```
replace: true,
```

```
});
```

```
};
```

```
return (
```

```
<div className="container mt-5">
```

```
<h1>Login</h1>
```

```
<hr />
```

```
<button className="btn btn-outline-primary" onClick={handleLogin}>
```

```
Login
```

```
</button>
```

```
</div>
```

```
);
```

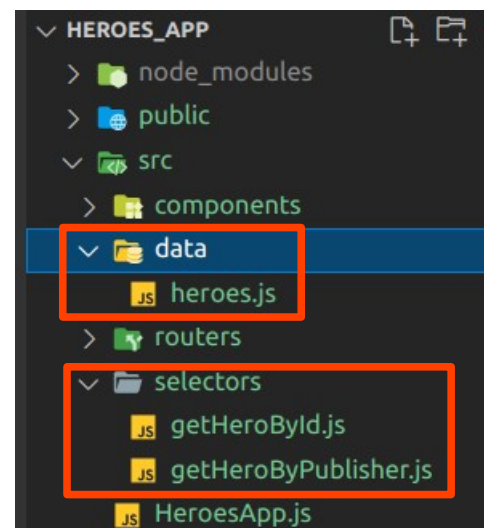
```
};
```

```
export default LoginScreen;
```

Lista de Heroes

Ver: <https://gist.github.com/Klerith/934da045caae0fec3a1067d013926c46>

Crear carpetas y archivos:



data:

```
export const heroes = [
```

```
{
```

```
id: "dc-batman",
```

```
superhero: "Batman",
```

```
publisher: "DC Comics",
```

```
alter_ego: "Bruce Wayne",
```

```
first_appearance: "Detective Comics #27",
```

```
characters: "Bruce Wayne",
```

```
},
```

```
{
```

```
id: "dc-superman",
```

```
superhero: "Superman",
```

```
publisher: "DC Comics",
```

```
alter_ego: "Kal-El",
```

```
first_appearance: "Action Comics #1",
```

```
characters: "Kal-El",
```

```
},
```

```
{
```

```
{
  id: "dc-flash",
  superhero: "Flash",
  publisher: "DC Comics",
  alter_ego: "Jay Garrick",
  first_appearance: "Flash Comics #1",
  characters: "Jay Garrick, Barry Allen, Wally West, Bart Allen",
},
{
  id: "dc-green",
  superhero: "Green Lantern",
  publisher: "DC Comics",
  alter_ego: "Alan Scott",
  first_appearance: "All-American Comics #16",
  characters: "Alan Scott, Hal Jordan, Guy Gardner, John Stewart, Kyle Raynor, Jade, Sinestro, Simon Baz",
},
{
  id: "dc-arrow",
  superhero: "Green Arrow",
  publisher: "DC Comics",
  alter_ego: "Oliver Queen",
  first_appearance: "More Fun Comics #73",
  characters: "Oliver Queen",
},
{
  id: "dc-wonder",
  superhero: "Wonder Woman",
  publisher: "DC Comics",
  alter_ego: "Princess Diana",
  first_appearance: "All Star Comics #8",
  characters: "Princess Diana",
},
{
  id: "dc-martian",
  superhero: "Martian Manhunter",
  publisher: "DC Comics",
  alter_ego: "J'onn J'onzz",
  first_appearance: "Detective Comics #225",
  characters: "Martian Manhunter",
},
{
  id: "dc-robin",
  superhero: "Robin/Nightwing",
  publisher: "DC Comics",
  alter_ego: "Dick Grayson",
  first_appearance: "Detective Comics #38",
  characters: "Dick Grayson",
},
{
  id: "dc-blue",
  superhero: "Blue Beetle",
  publisher: "DC Comics",
  alter_ego: "Dan Garret",
  first_appearance: "Mystery Men Comics #1",
  characters: "Dan Garret, Ted Kord, Jaime Reyes",
},
{
  id: "dc-black",
  superhero: "Black Canary",
  publisher: "DC Comics",
  alter_ego: "Dinah Drake",
  first_appearance: "Flash Comics #86",
  characters: "Dinah Drake, Dinah Lance",
},
{
  id: "marvel-spider",
  superhero: "Spider Man",
  publisher: "Marvel Comics",
  alter_ego: "Peter Parker",
  first_appearance: "Amazing Fantasy #15",
}
```

```
characters: "Peter Parker",
},
{
id: "marvel-captain",
superhero: "Captain America",
publisher: "Marvel Comics",
alter_ego: "Steve Rogers",
first_appearance: "Captain America Comics #1",
characters: "Steve Rogers",
},
{
id: "marvel-iron",
superhero: "Iron Man",
publisher: "Marvel Comics",
alter_ego: "Tony Stark",
first_appearance: "Tales of Suspense #39",
characters: "Tony Stark",
},
{
id: "marvel-thor",
superhero: "Thor",
publisher: "Marvel Comics",
alter_ego: "Thor Odinson",
first_appearance: "Journey into Myster #83",
characters: "Thor Odinson",
},
{
id: "marvel-hulk",
superhero: "Hulk",
publisher: "Marvel Comics",
alter_ego: "Bruce Banner",
first_appearance: "The Incredible Hulk #1",
characters: "Bruce Banner",
},
{
id: "marvel-wolverine",
superhero: "Wolverine",
publisher: "Marvel Comics",
alter_ego: "James Howlett",
first_appearance: "The Incredible Hulk #180",
characters: "James Howlett",
},
{
id: "marvel-daredevil",
superhero: "Daredevil",
publisher: "Marvel Comics",
alter_ego: "Matthew Michael Murdock",
first_appearance: "Daredevil #1",
characters: "Matthew Michael Murdock",
},
{
id: "marvel-hawkeye",
superhero: "Hawkeye",
publisher: "Marvel Comics",
alter_ego: "Clinton Francis Barton",
first_appearance: "Tales of Suspense #57",
characters: "Clinton Francis Barton",
},
{
id: "marvel-cyclops",
superhero: "Cyclops",
publisher: "Marvel Comics",
alter_ego: "Scott Summers",
first_appearance: "X-Men #1",
characters: "Scott Summers",
},
{
id: "marvel-silver",
superhero: "Silver Surfer",
publisher: "Marvel Comics",
```

```
alter_ego: "Norrin Radd",
first_appearance: "The Fantastic Four #48",
characters: "Norrin Radd",
},
];
```

getHeroByPublisher:

```
import { heroes } from "../data/heroes";

export const getHeroByPublisher = (publisher) => {
const validPublisher = ["DC Comics", "Marvel Comics"];
if (!validPublisher.includes(publisher)) {
throw new Error(`${publisher} is not a valid publisher`);
}
return heroes.filter((hero) => hero.publisher === publisher);
};
```

HeroList:

```
import { getHeroByPublisher } from "../selectors/getHeroByPublisher";
```

```
const HeroList = ({ publisher }) => {
const heroes = getHeroByPublisher(publisher);
```

```
return (
<>
<ul>
{heroes.map((hero) => (
<li key={hero.id}> {hero.superhero} </li>
))}
</ul>
</>
);
};
```

```
export default HeroList;
```

DcScreen:

```
import HeroList from "../hero/HeroList";
```

```
const DcScreen = () => {
return (
<div>
<h1>DcScreen</h1>
<hr />
<HeroList publisher="DC Comics"/>
</div>
);
};
```

```
export default DcScreen;
```

MarvelScreen:

```
import HeroList from "../hero/HeroList";
```

```
const MarvelScreen = () => {
return (
<div>
<h1>MarvelScreen</h1>
<hr />
<HeroList publisher="Marvel Comics" />
</div>
);
};
```

```
export default MarvelScreen;
```

Tarjetas con la información del Héroe

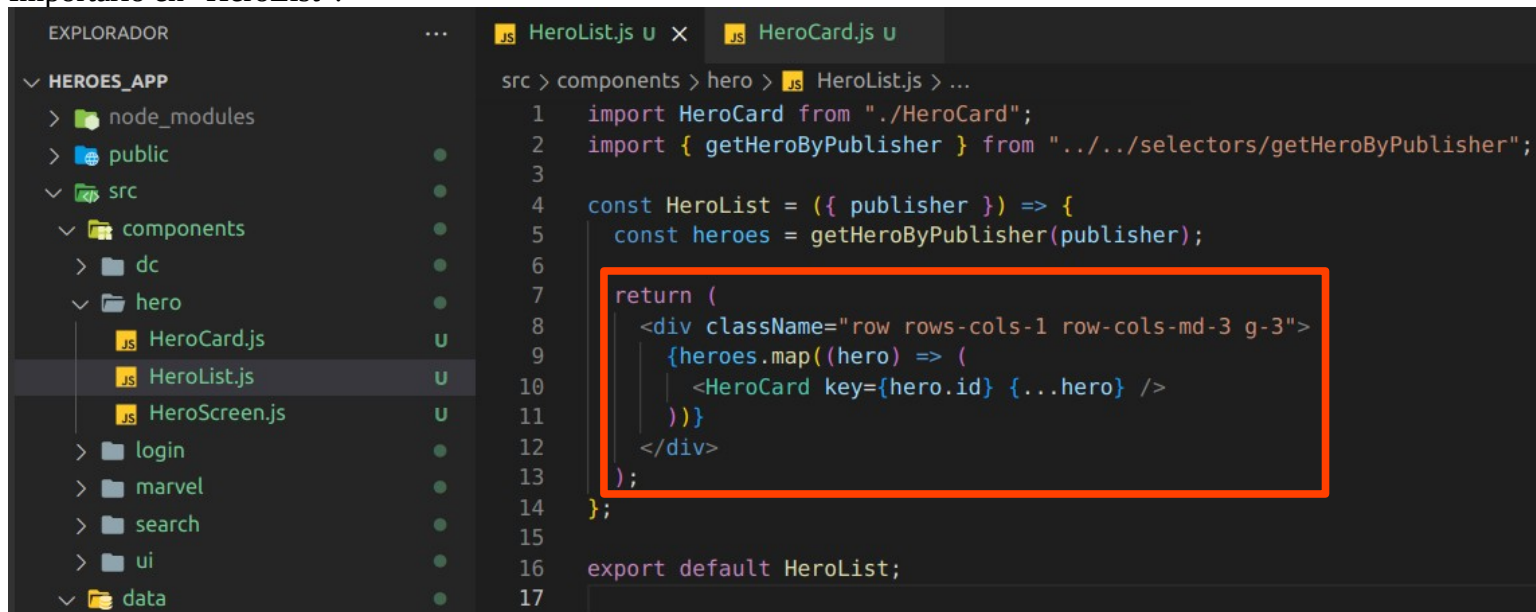
Crear componente “HeroCard”:

```
import { Link } from "react-router-dom";

const HeroCard = ({
  id,
  superhero,
  publisher,
  alter_ego,
  first_appearance,
  characters,
}) => {
  const imagePath = `/assets/${id}.jpg`;
  return (
    <div className="col">
      <div className="card">
        <div className="row no-gutters">
          <div className="col-4">
            <img src={imagePath} className="card-img" alt={superhero} />
          </div>
          <div className="col-8">
            <div className="card-body">
              <h5 className="card-title"> {superhero} </h5>
              <p className="card-text"> {alter_ego} </p>
              {alter_ego !== characters && (
                <p className="text-muted"> {characters} </p>
              )}
              <p className="card-text">
                <small className="text-muted"> {first_appearance} </small>
              </p>
              <Link to={`/hero/${id}`}>More...</Link>
            </div>
          </div>
        </div>
      </div>
    </div>
  );
};

export default HeroCard;
```

Importarlo en “HeroList”:

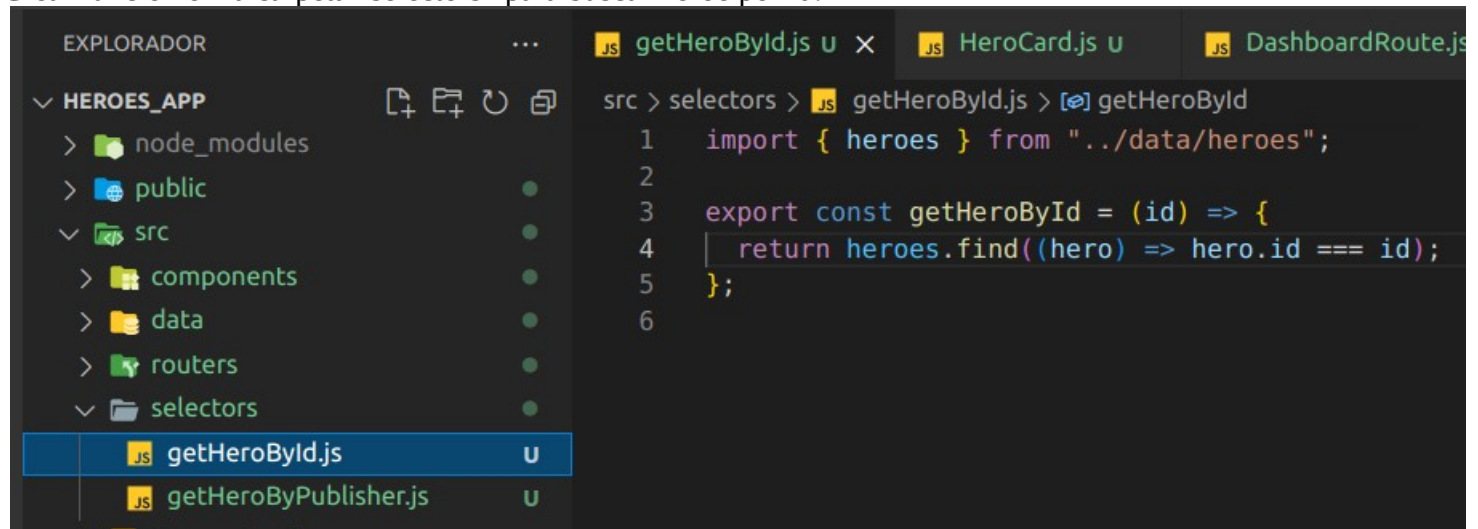


The screenshot shows a code editor with two files open: HeroList.js and HeroCard.js. The Explorer sidebar on the left shows the project structure for 'HEROES_APP', including folders like 'node_modules', 'public', 'src', and 'components', and files like 'HeroCard.js', 'HeroList.js', and 'HeroScreen.js'. The main editor displays the content of HeroList.js, which imports HeroCard and uses it in a map function to render a list of heroes. The map function is highlighted with a red box.

```
src > components > hero > JS HeroList.js > ...
1  import HeroCard from "../HeroCard";
2  import { getHeroByPublisher } from "../../selectors/getHeroByPublisher";
3
4  const HeroList = ({ publisher }) => {
5    const heroes = getHeroByPublisher(publisher);
6
7    return (
8      <div className="row rows-cols-1 row-cols-md-3 g-3">
9        {heroes.map((hero) => (
10          <HeroCard key={hero.id} {...hero} />
11        ))}
12      </div>
13    );
14  };
15
16  export default HeroList;
17
```


Leer argumentos por URL

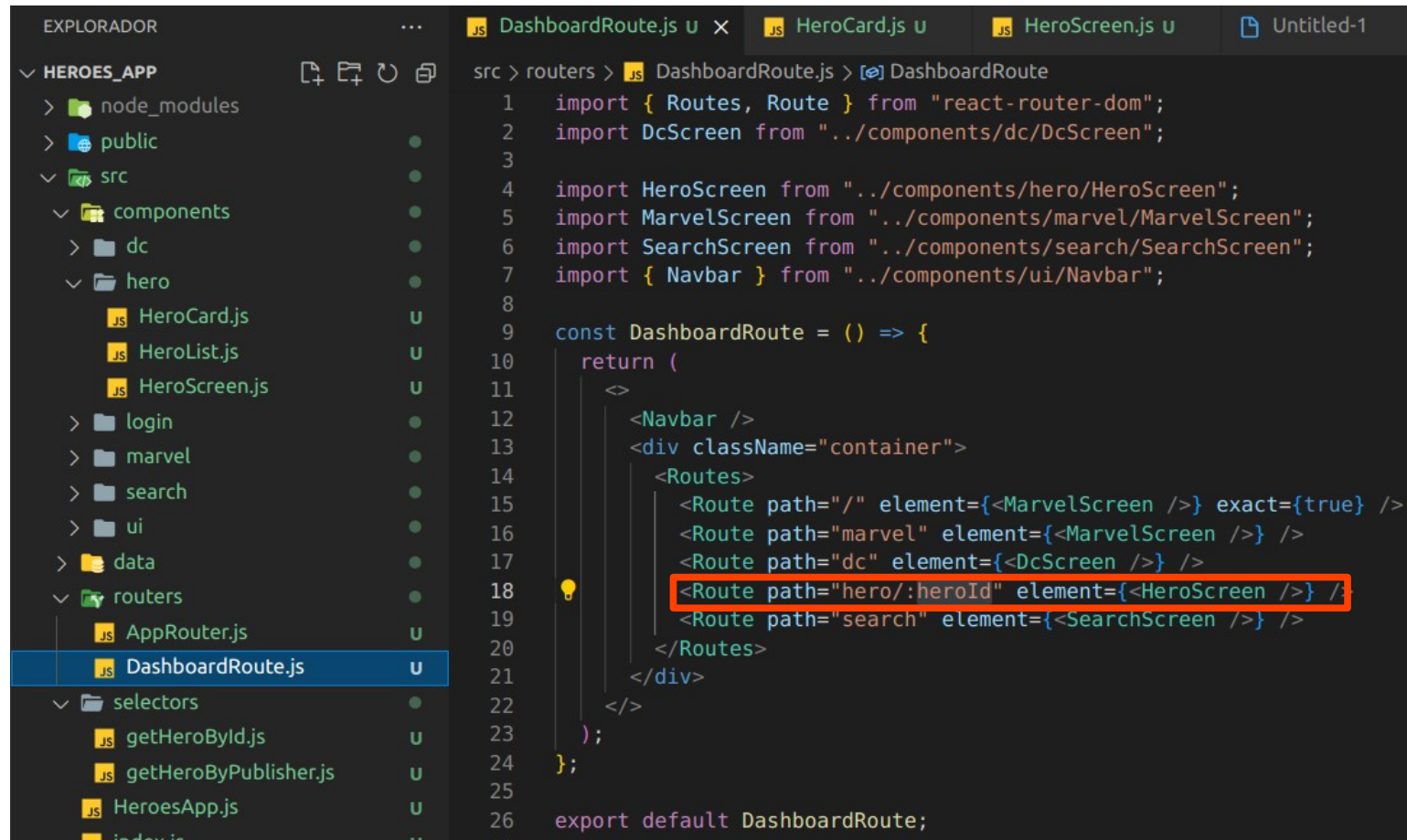
Crear función en la carpeta “selectors” para buscar héroe por id:



The screenshot shows the VS Code interface. On the left, the 'EXPLORADOR' (Explorer) sidebar shows the project structure with the 'selectors' folder expanded. The file 'getHeroById.js' is selected. On the right, the editor shows the code for 'getHeroById.js'.

```
src > selectors > JS getHeroById.js > [?] getHeroById
1  import { heroes } from "../data/heroes";
2
3  export const getHeroById = (id) => {
4    return heroes.find((hero) => hero.id === id);
5  };
6
```

Crear la ruta en “DashboadRoute”:



The screenshot shows the VS Code interface. On the left, the 'EXPLORADOR' (Explorer) sidebar shows the project structure with the 'routers' folder expanded. The file 'DashboardRoute.js' is selected. On the right, the editor shows the code for 'DashboardRoute.js'.

```
src > routers > JS DashboardRoute.js > [?] DashboardRoute
1  import { Routes, Route } from "react-router-dom";
2  import DcScreen from "../components/dc/DcScreen";
3
4  import HeroScreen from "../components/hero/HeroScreen";
5  import MarvelScreen from "../components/marvel/MarvelScreen";
6  import SearchScreen from "../components/search/SearchScreen";
7  import { Navbar } from "../components/ui/Navbar";
8
9  const DashboardRoute = () => {
10    return (
11      <>
12        <Navbar />
13        <div className="container">
14          <Routes>
15            <Route path="/" element={<MarvelScreen />} exact={true} />
16            <Route path="marvel" element={<MarvelScreen />} />
17            <Route path="dc" element={<DcScreen />} />
18            <Route path="hero/:heroId" element={<HeroScreen />} />
19            <Route path="search" element={<SearchScreen />} />
20          </Routes>
21        </div>
22      </>
23    );
24  };
25
26  export default DashboardRoute;
```


Recordar que componente HeroCard llama a la url con el parámetro “id”:



```
import { useParams, Navigate, useNavigate } from "react-router-dom";
import { getHeroById } from "../../selectors/getHeroById";
```

```
const handleReturn = () => {
  navigate(-1); // Ir a página anterior
};
```

```

return (
<div className="row mt-5">
<div className="col-4">
<img src={imagePath} className="img-thumbnail" alt={superhero} />
</div>
<div className="col-8">
<h3> {superhero} </h3>
<ul className="list-group list-group-flush">
<li className="list-group-item">
<b>Alter Ego:</b> {alter_ego}
</li>
<li className="list-group-item">
<b>Publisher:</b> {publisher}
</li>
<li className="list-group-item">
<b>First Appearance:</b> {first_appearance}
</li>
</ul>
<h5 className="mt-3">Characteres</h5>
<p> {characters} </p>
<button className="btn btn-outline-info" onClick={handleReturn}>
Return
</button>
</div>
</div>

```

```
export default HeroScreen;
```

Nota useMemo

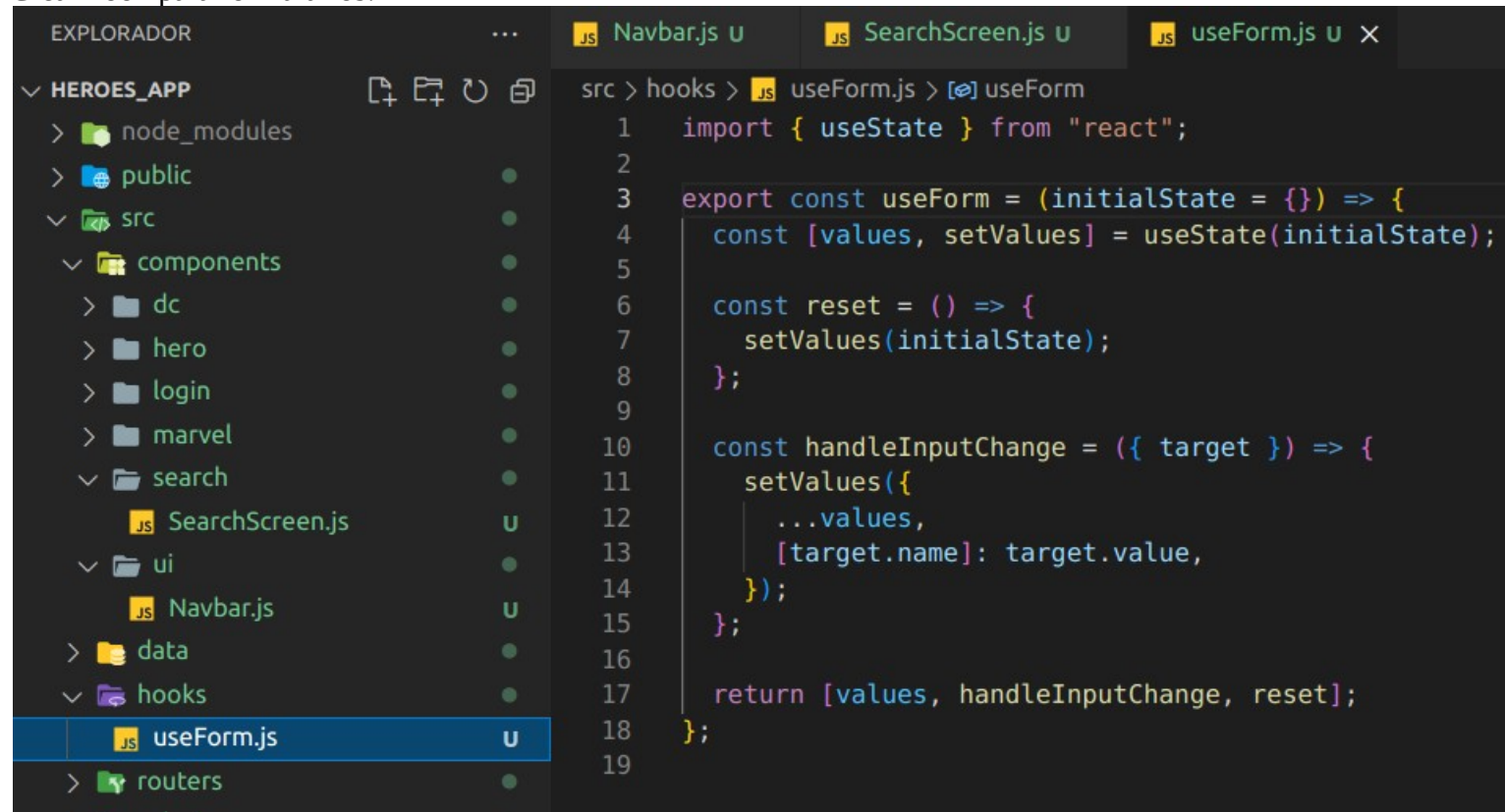
Usar hook “useMemo” para optimizar funciones “getHeroById” y “getHeroByPublisher”:

```
JS HeroScreen.js X JS HeroList.js U JS getHeroById.js U
src > components > hero > JS HeroScreen.js > [e] HeroScreen
1  import { useMemo } from "react";
2  import { useParams, Navigate, useNavigate } from "react-router-dom";
3  import { getHeroById } from "../../selectors/getHeroById";
4
5  const HeroScreen = () => {
6    // Hook para leer los parámetros de la url
7    const { heroId } = useParams(); // De HeroCard
8    // hook para regresar a página anterior
9    const navigate = useNavigate();
10
11    /* getHeroById debe estar optimizada (en memoria), ya que se llama
12     cuando hay algún cambio en componente. Solo disparar función cuando
13     "heroId" cambie */
14    const hero = useMemo(() => getHeroById(heroId), [heroId]);
15
16    if (!hero) {
17      return <Navigate to="/" />;
18    }
19
20    // Desestructurando hero
21    const { id, superhero, publisher, alter_ego, first_appearance, characters } =
22      hero;
23
24    const imagePath = `/assets/${id}.jpg`;
25
26    const handleReturn = () => {
27      navigate(-1); // Ir a página anterior
28    };
29  };
30
```

```
JS HeroList.js X JS getHeroByPublisher.js U
src > components > hero > JS HeroList.js > [e] HeroList > [e] heroes
1  import HeroCard from "../HeroCard";
2  import { getHeroByPublisher } from "../../selectors/getHeroByPublisher";
3  import { useMemo } from "react";
4
5  const HeroList = ({ publisher }) => {
6
7    /* getHeroByPublisher debe estar optimizada (en memoria), ya que se llama
8     cuando hay algún cambio en componente. Solo disparar función cuando
9     "publisher" cambie */
10    const heroes = useMemo(() => getHeroByPublisher(publisher), [publisher]);
11
12    return (
13      <div className="row rows-cols-1 row-cols-md-3 g-3">
14        {heroes.map((hero) => (
15          <HeroCard key={hero.id} {...hero} />
16        ))}
17      </div>
18    );
19  };
20
```

SearchComponent

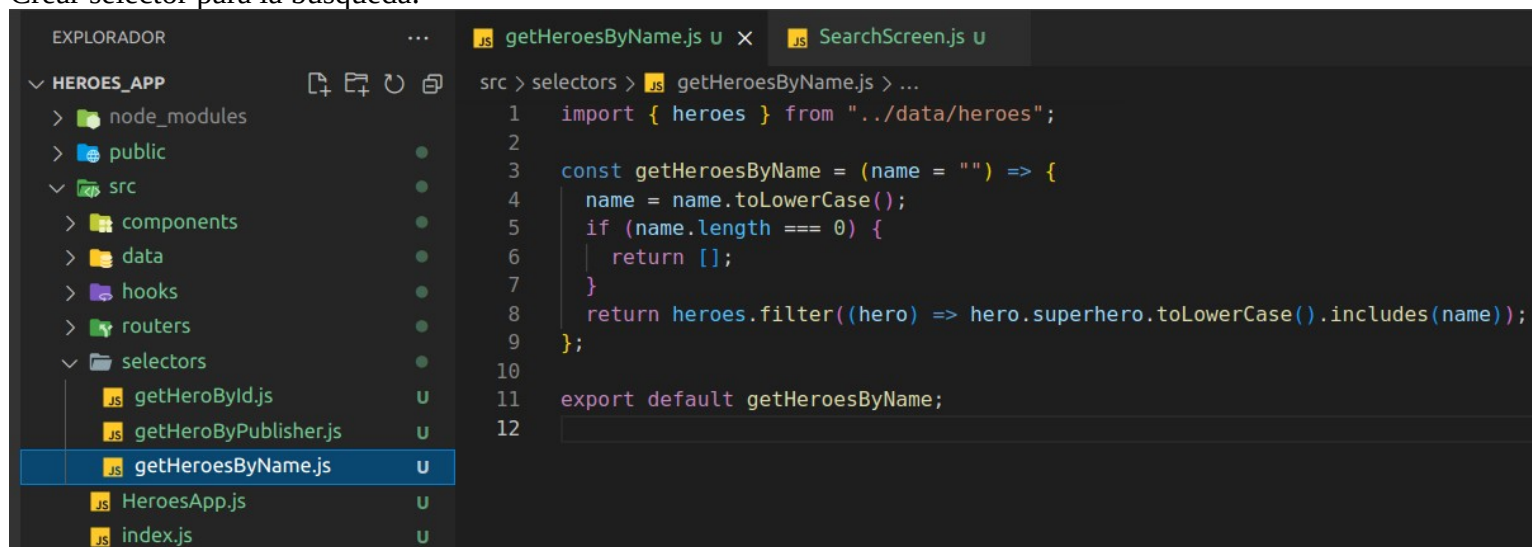
Crear hook para formularios:



```
EXPLORADOR
HEROES_APP
  node_modules
  public
  src
    components
    dc
    hero
    login
    marvel
    search
      SearchScreen.js
    ui
      Navbar.js
    data
    hooks
      useForm.js
    routers

src > hooks > useForm.js > useForm
1  import { useState } from "react";
2
3  export const useForm = (initialState = {}) => {
4    const [values, setValues] = useState(initialState);
5
6    const reset = () => {
7      setValues(initialState);
8    };
9
10   const handleInputChange = ({ target }) => {
11     setValues({
12       ...values,
13       [target.name]: target.value,
14     });
15   };
16
17   return [values, handleInputChange, reset];
18 };
19
```

Crear selector para la búsqueda:



```
EXPLORADOR
HEROES_APP
  node_modules
  public
  src
    components
    data
    hooks
    routers
    selectors
      getHeroById.js
      getHeroByPublisher.js
      getHeroesByName.js
      HeroesApp.js
      index.js

src > selectors > getHeroesByName.js > ...
1  import { heroes } from "../data/heroes";
2
3  const getHeroesByName = (name = "") => {
4    name = name.toLowerCase();
5    if (name.length === 0) {
6      return [];
7    }
8    return heroes.filter((hero) => hero.superhero.toLowerCase().includes(name));
9  };
10
11  export default getHeroesByName;
12
```

En "SearchScreen.js":

Usar el paquete query-string para trabajar con los parámetros que vienen en la ruta:

<https://www.npmjs.com/package/query-string>

```
import { useMemo } from "react";
import { useLocation, useNavigate } from "react-router-dom";
```

```
import { useForm } from "../../hooks/useForm";
import getHeroesByName from "../../selectors/getHeroesByName";
import HeroCard from "../hero/HeroCard";
const queryString = require("query-string");
```

```
const SearchScreen = () => {
  // navigate para los parámetros de búsqueda
  const navigate = useNavigate();
  // Para saber la localización (url) || Usar query-string (https://www.npmjs.com/package/query-string)
  const location = useLocation();
  // query
  const { q = "" } = queryString.parse(location.search);

  // Hook para el formulario
```

```

const initialForm = {
  searchText: q,
};

const [formValues, handleInputChange] = useForm(initialForm);
const { searchText } = formValues;
// Héroes filtrados por "superHero" (useMemo para optimizar)
const heroesFilter = useMemo(() => getHeroesByName(q), [q]);

const handleSearch = (event) => {
  event.preventDefault();
  navigate(`?q=${searchText}`); // enviar un queryParam
};

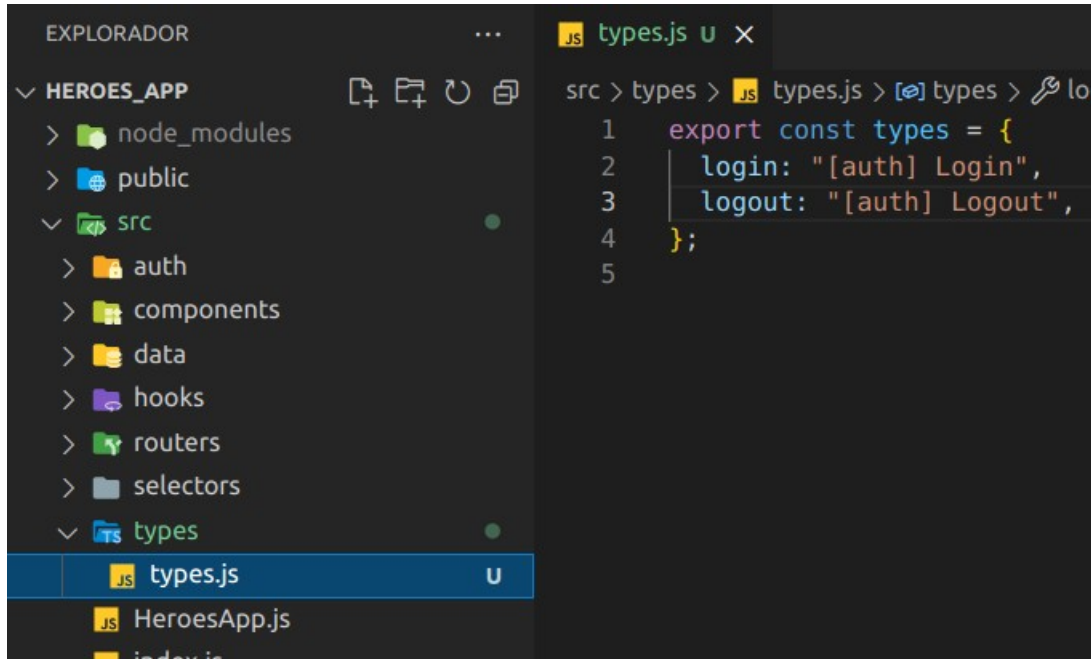
return (
  <div>
    <h1>Search</h1>
    <hr />
    <div className="row">
      <div className="col-5">
        <h4>Search</h4>
        <hr />
        <form onSubmit={handleSearch}>
          <input
            type="text"
            placeholder="Hero"
            className="form-control"
            name="searchText"
            autoComplete="off"
            value={searchText}
            onChange={handleInputChange}
          />
          <button type="submit" className="btn btn-outline-info mt-1">
            Search...
          </button>
        </form>
      </div>
      <div className="col-7">
        <h4>Resultado</h4>
        <hr />
        {q === "" ? (
          <div className="alert alert-info">Buscar un héroe</div>
        ) : heroesFilter.length === 0 ? (
          <div className="alert alert-danger">
            No se encontraron coincidencias con {q}
          </div>
        ) : (
          heroesFilter.map((hero) => <HeroCard key={hero.id} {...hero} />)
        )}
      </div>
    </div>
  );
};
export default SearchScreen;

```


Protección de rutas

Context y Reducer de mi aplicación

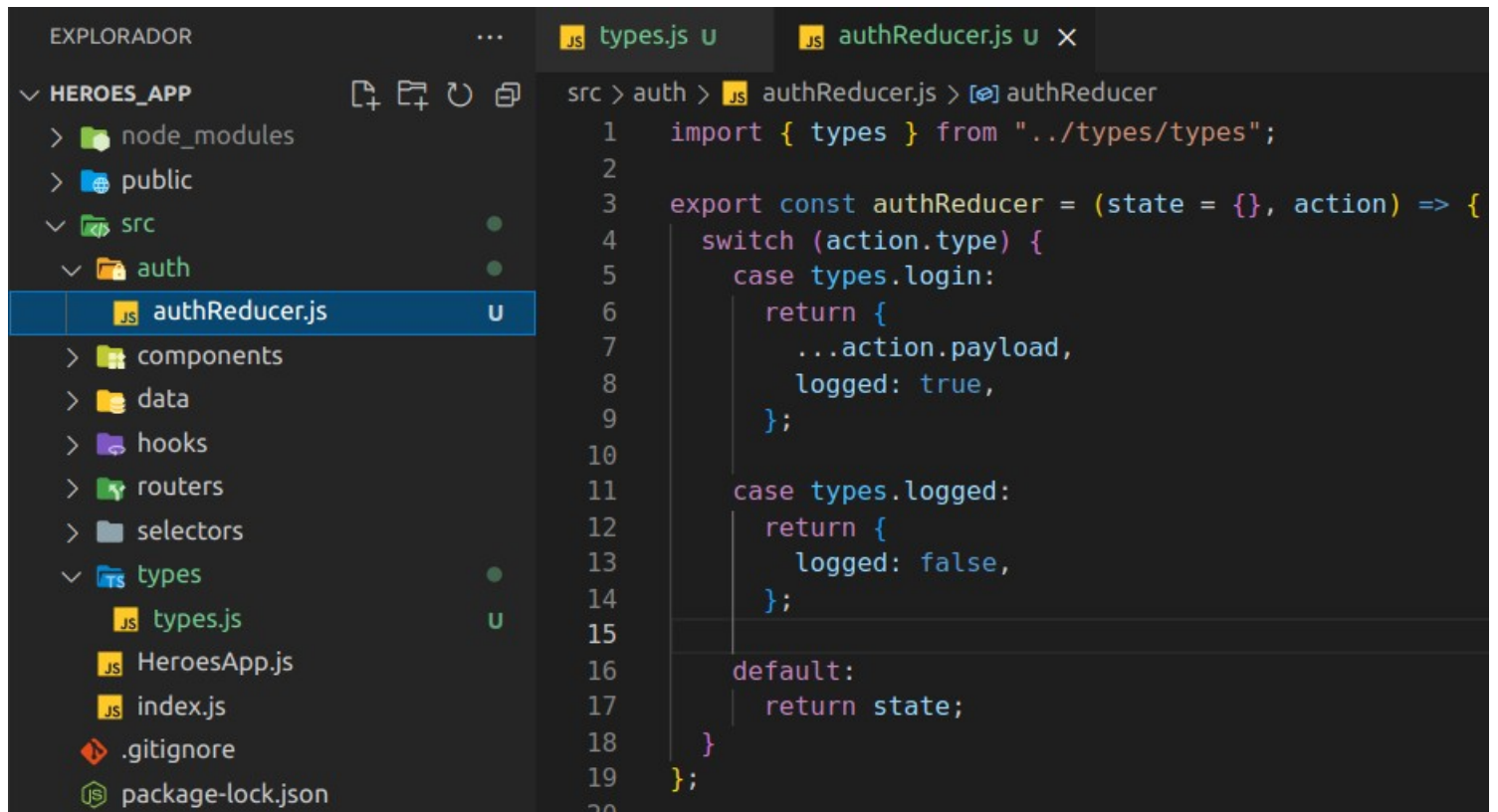
Debe haber un estado global de la aplicación para saber si el usuario está autenticado. En la carpeta “types”, crear “types.js” (que son las acciones a disparar)



```
EXPLORADOR
HEROES_APP
  node_modules
  public
  src
    auth
    components
    data
    hooks
    routers
    selectors
    types
      types.js
    HeroesApp.js
    index.js

src > types > types.js
1 export const types = {
2   login: "[auth] Login",
3   logout: "[auth] Logout",
4 };
5
```

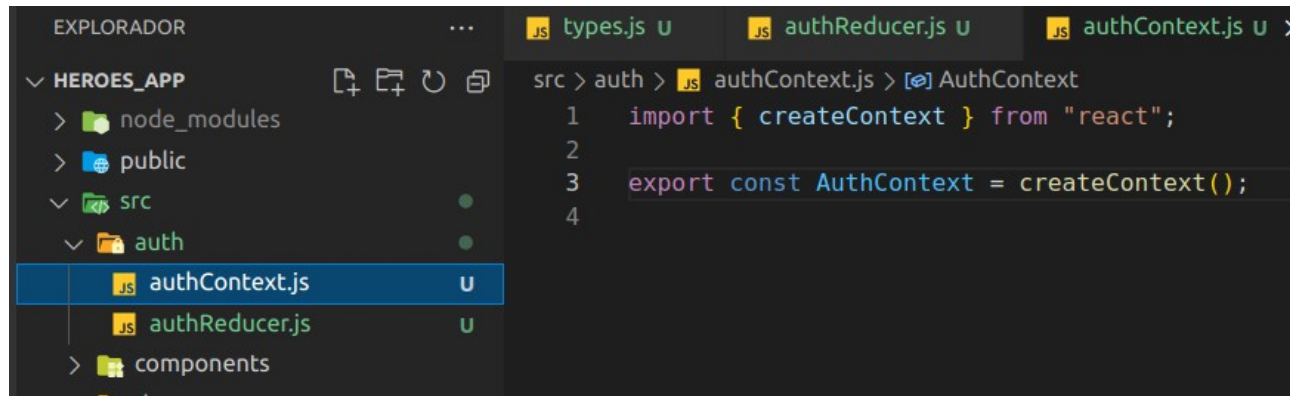
Crear un reducer en “auth”:



```
EXPLORADOR
HEROES_APP
  node_modules
  public
  src
    auth
      authReducer.js
    components
    data
    hooks
    routers
    selectors
    types
      types.js
    HeroesApp.js
    index.js
  .gitignore
  package-lock.json

src > auth > authReducer.js
1 import { types } from "../types/types";
2
3 export const authReducer = (state = {}, action) => {
4   switch (action.type) {
5     case types.login:
6       return {
7         ...action.payload,
8         logged: true,
9       };
10
11     case types.logged:
12       return {
13         logged: false,
14       };
15
16     default:
17       return state;
18   }
19 };
20
```

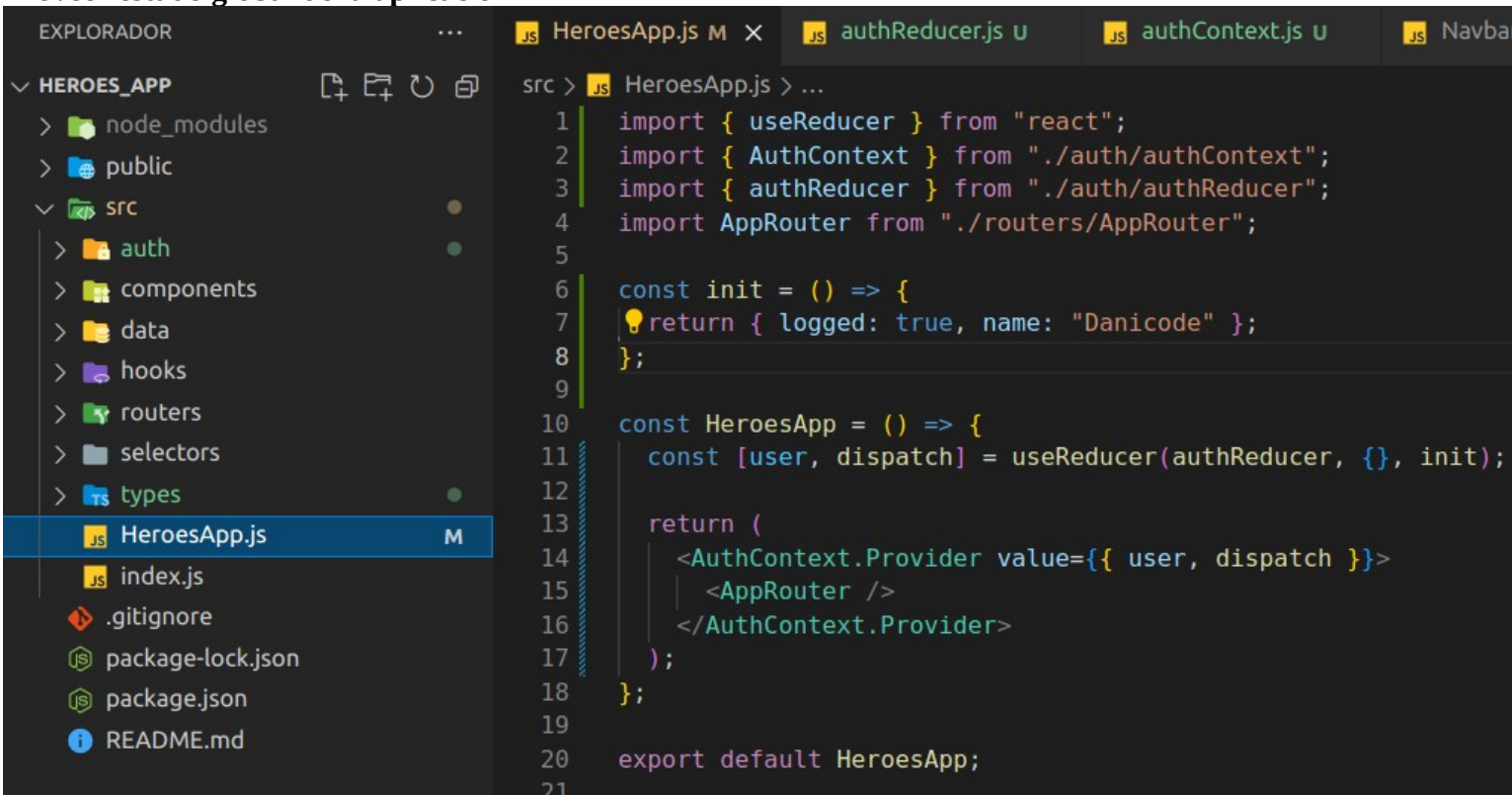
Además, crear el contexto para administrar el reducer anterior; este contexto proveerá información a todos los componente hijos (se colocará el manejo del contexto en componente AppRouter.js, que está en la parte más alta, luego del index.js, de todos los componentes).



The screenshot shows the VS Code interface. On the left, the 'EXPLORADOR' (Explorer) sidebar shows the project structure: 'HEROES_APP' > 'src' > 'auth'. The 'authContext.js' file is selected. The main editor shows the content of 'authContext.js':

```
src > auth > authContext.js > [?] AuthContext
1  import { createContext } from "react";
2
3  export const AuthContext = createContext();
4
```

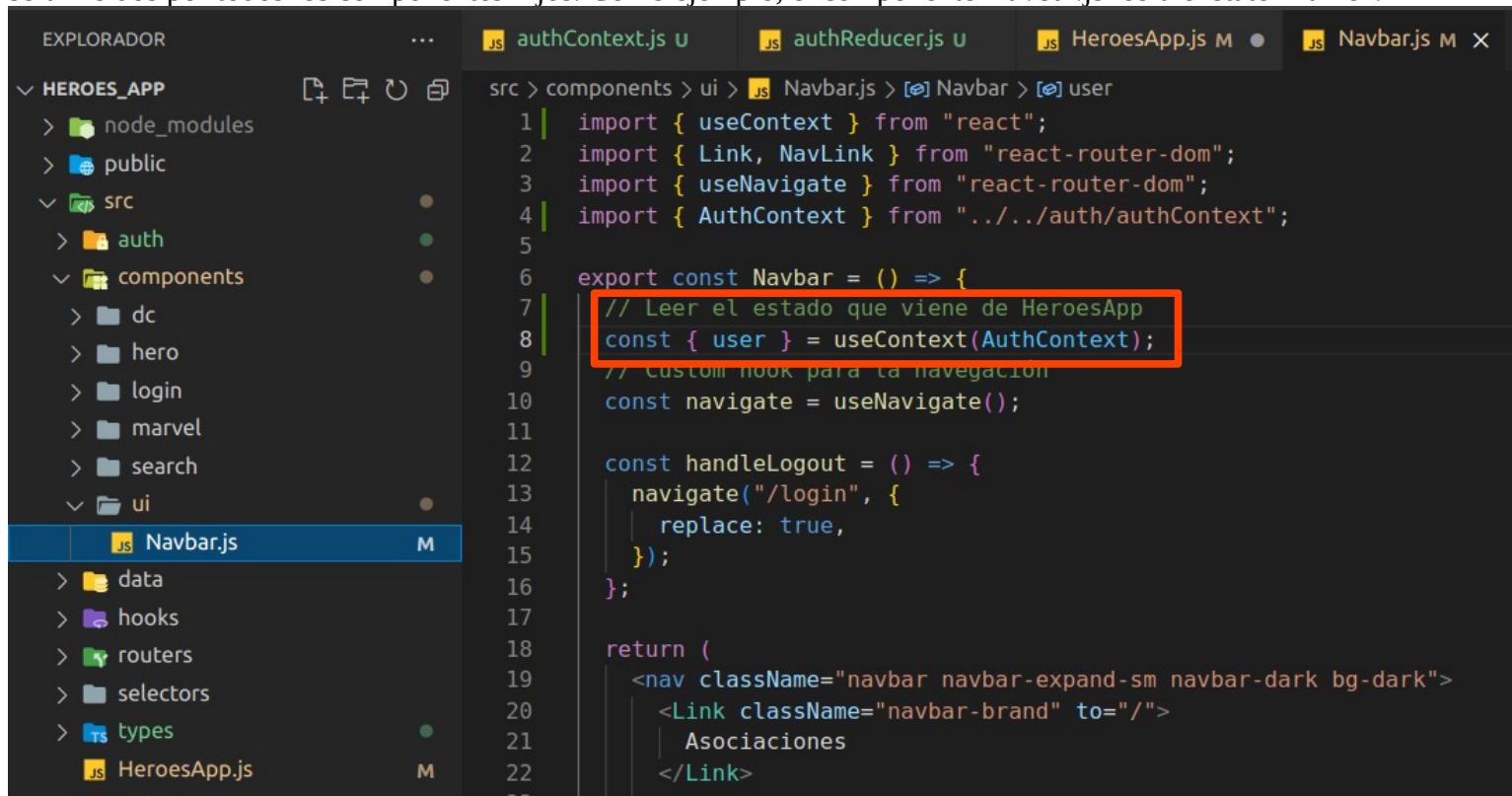
Proveer estado global de la aplicación



The screenshot shows the VS Code interface. On the left, the 'EXPLORADOR' (Explorer) sidebar shows the project structure: 'HEROES_APP' > 'src' > 'HeroesApp.js' is selected. The main editor shows the content of 'HeroesApp.js':

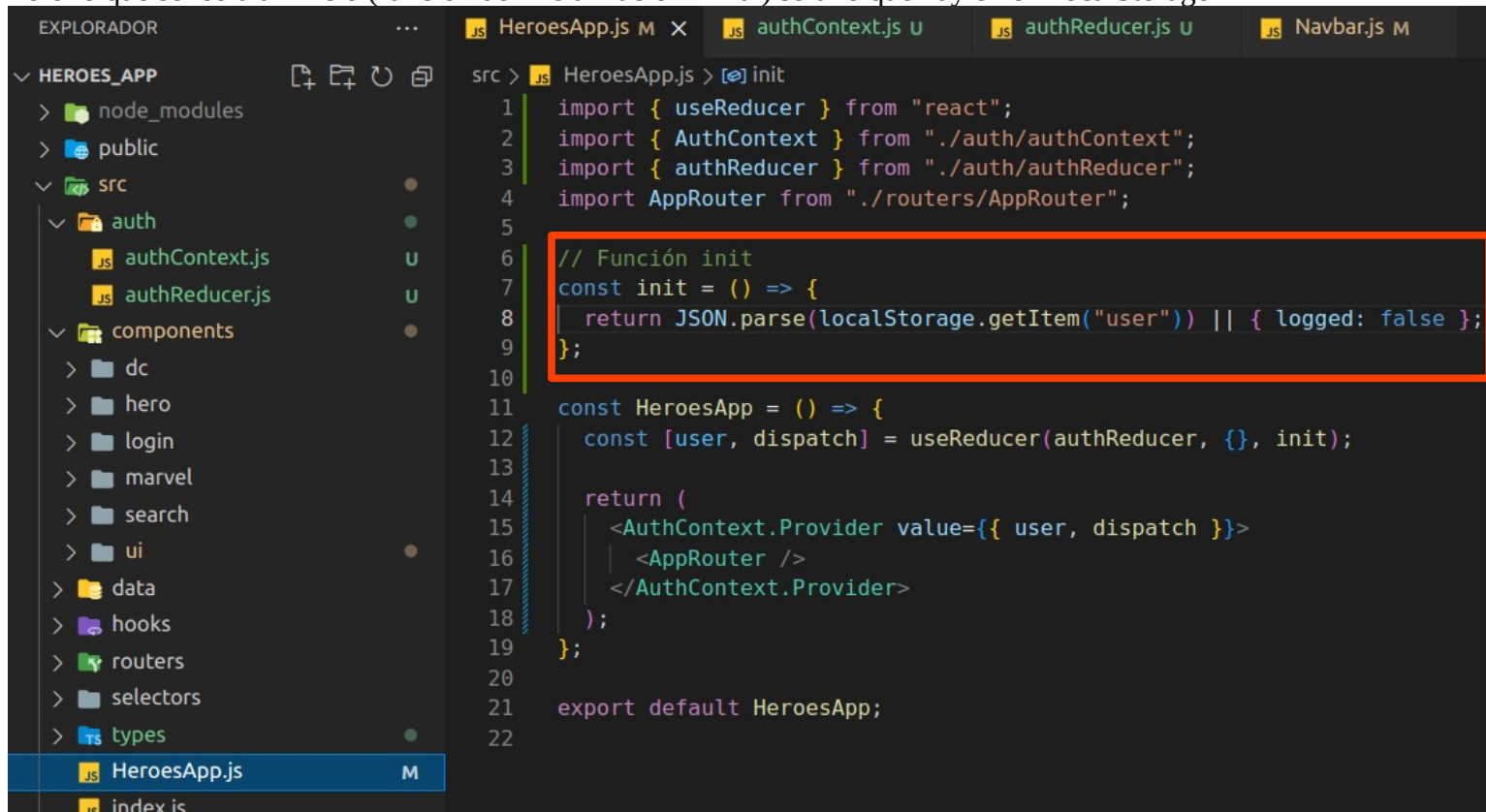
```
src > JS HeroesApp.js > ...
1  import { useReducer } from "react";
2  import { AuthContext } from "../auth/authContext";
3  import { authReducer } from "../auth/authReducer";
4  import AppRouter from "../routers/AppRouter";
5
6  const init = () => {
7    return { logged: true, name: "Danicode" };
8  };
9
10 const HeroesApp = () => {
11   const [user, dispatch] = useReducer(authReducer, {}, init);
12
13   return (
14     <AuthContext.Provider value={{ user, dispatch }}>
15       <AppRouter />
16     </AuthContext.Provider>
17   );
18 };
19
20 export default HeroesApp;
21
```

Como ejemplo, la función inicial (init) envía los parámetros “logged”: true y “name”: “Danicode”. Estos parámetros serán leídos por todos los componentes hijos. Como ejemplo, el componente Navbar.js leerá el state “name”:



```
EXPLORADOR src > components > ui > JS Navbar.js > [x] Navbar > [x] user
1 | import { useContext } from "react";
2 | import { Link, NavLink } from "react-router-dom";
3 | import { useNavigate } from "react-router-dom";
4 | import { AuthContext } from "../../auth/authContext";
5 |
6 | export const Navbar = () => {
7 |   // Leer el estado que viene de HeroesApp
8 |   const { user } = useContext(AuthContext);
9 |   // Custom hook para la navegación
10 |   const navigate = useNavigate();
11 |
12 |   const handleLogout = () => {
13 |     navigate("/login", {
14 |       replace: true,
15 |     });
16 |   };
17 |
18 |   return (
19 |     <nav className="navbar navbar-expand-sm navbar-dark bg-dark">
20 |       <Link className="navbar-brand" to="/">
21 |         Asociaciones
22 |       </Link>
```

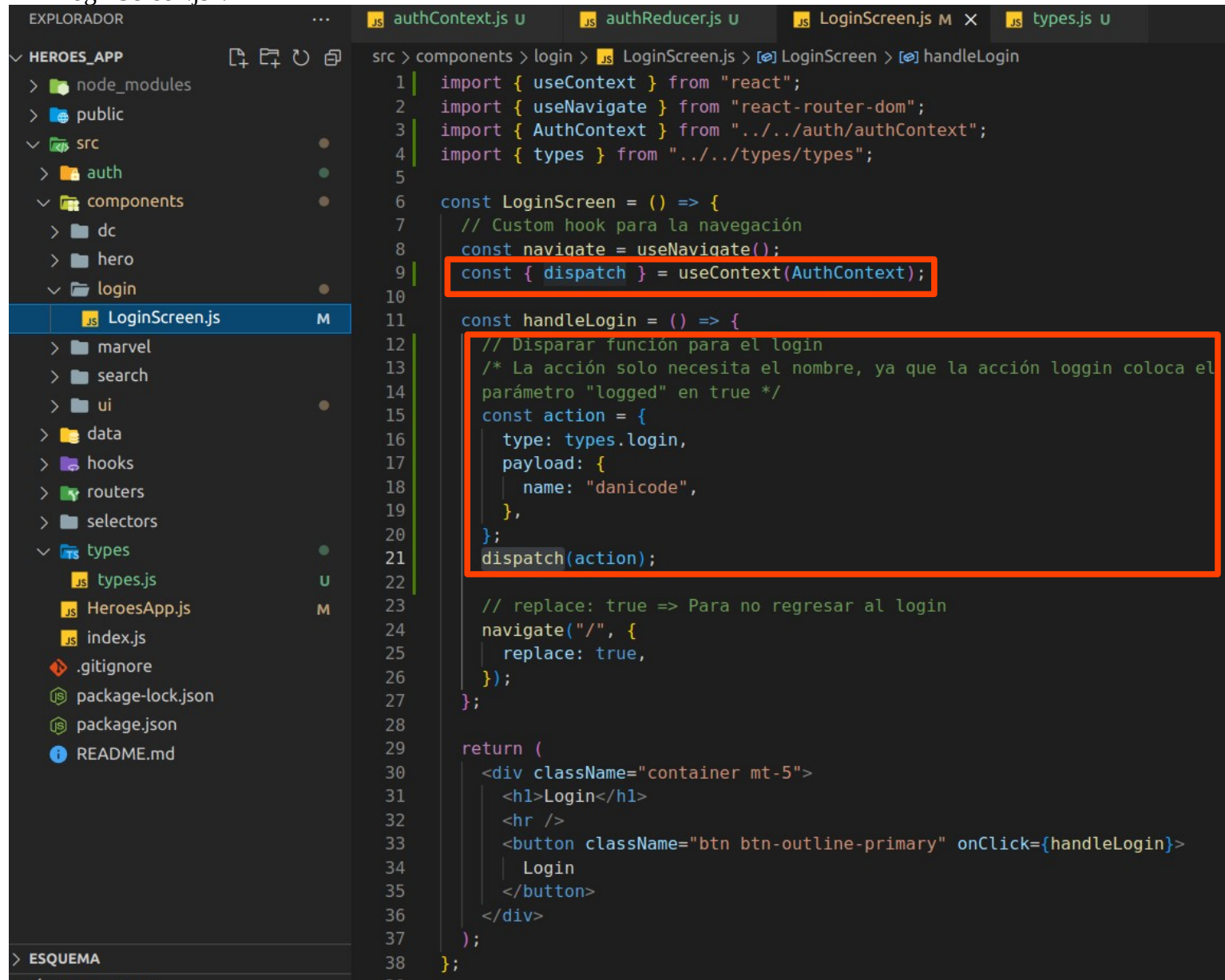
Pero lo que se leerá al inicio (función de inicialización “init”) será lo que hay en el “localStorage”



```
EXPLORADOR src > JS HeroesApp.js > [x] init
1 | import { useReducer } from "react";
2 | import { AuthContext } from "../auth/authContext";
3 | import { authReducer } from "../auth/authReducer";
4 | import AppRouter from "../routers/AppRouter";
5 |
6 | // Función init
7 | const init = () => {
8 |   return JSON.parse(localStorage.getItem("user")) || { logged: false };
9 | };
10 |
11 | const HeroesApp = () => {
12 |   const [user, dispatch] = useReducer(authReducer, {}, init);
13 |
14 |   return (
15 |     <AuthContext.Provider value={{ user, dispatch }}>
16 |       <AppRouter />
17 |     </AuthContext.Provider>
18 |   );
19 | };
20 |
21 | export default HeroesApp;
```

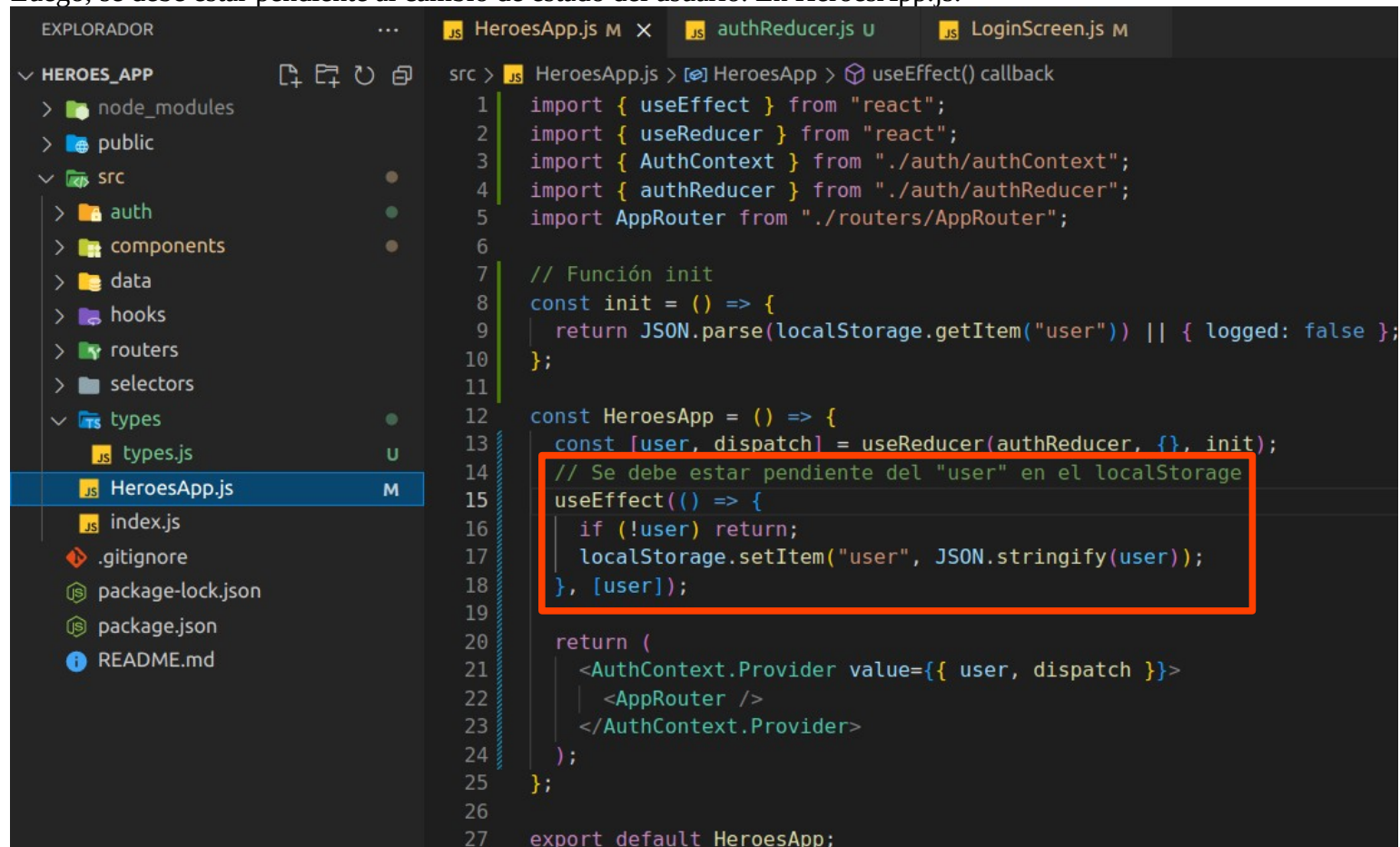

Login de un usuario

En "LoginScreen.js":



```
src > components > login > JS LoginScreen.js > [e] LoginScreen > [e] handleLogin
1 | import { useContext } from "react";
2 | import { useNavigate } from "react-router-dom";
3 | import { AuthContext } from "../../auth/authContext";
4 | import { types } from "../../types/types";
5 |
6 | const LoginScreen = () => {
7 |   // Custom hook para la navegación
8 |   const navigate = useNavigate();
9 |   const { dispatch } = useContext(AuthContext);
10 |
11 |   const handleLogin = () => {
12 |     // Disparar función para el login
13 |     /* La acción solo necesita el nombre, ya que la acción loggin coloca el
14 |      parámetro "logged" en true */
15 |     const action = {
16 |       type: types.login,
17 |       payload: {
18 |         name: "danicode",
19 |       },
20 |     };
21 |     dispatch(action);
22 |
23 |     // replace: true => Para no regresar al login
24 |     navigate("/", {
25 |       replace: true,
26 |     });
27 |   };
28 |
29 |   return (
30 |     <div className="container mt-5">
31 |       <h1>Login</h1>
32 |       <hr />
33 |       <button className="btn btn-outline-primary" onClick={handleLogin}>
34 |         Login
35 |       </button>
36 |     </div>
37 |   );
38 | };
39 |
```

Luego, se debe estar pendiente al cambio de estado del usuario. En HeroesApp.js:



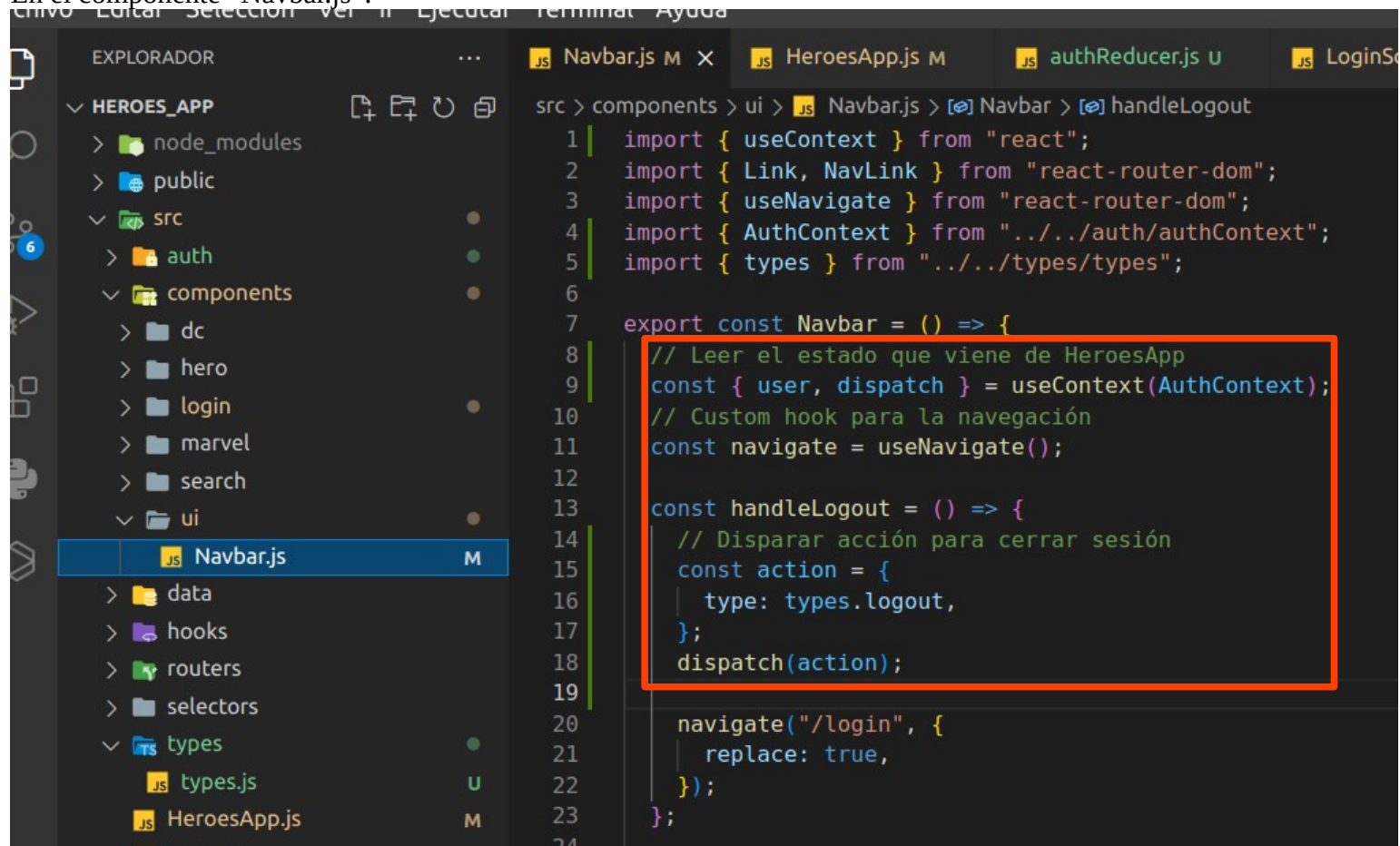
```
EXPLORADOR  ...  JS HeroesApp.js M  JS authReducer.js U  JS LoginScreen.js M

HEROES_APP
  node_modules
  public
  src
    auth
    components
    data
    hooks
    routers
    selectors
    types
      types.js
    JS HeroesApp.js M
    JS index.js
  .gitignore
  package-lock.json
  package.json
  README.md

src > JS HeroesApp.js > HeroesApp > useEffect() callback
1  import { useEffect } from "react";
2  import { useReducer } from "react";
3  import { AuthContext } from "../auth/authContext";
4  import { authReducer } from "../auth/authReducer";
5  import AppRouter from "../routers/AppRouter";
6
7  // Función init
8  const init = () => {
9    return JSON.parse(localStorage.getItem("user")) || { logged: false };
10 };
11
12 const HeroesApp = () => {
13   const [user, dispatch] = useReducer(authReducer, {}, init);
14   // Se debe estar pendiente del "user" en el localStorage
15   useEffect(() => {
16     if (!user) return;
17     localStorage.setItem("user", JSON.stringify(user));
18   }, [user]);
19
20   return (
21     <AuthContext.Provider value={{ user, dispatch }}>
22       <AppRouter />
23     </AuthContext.Provider>
24   );
25 };
26
27 export default HeroesApp;
```

Logout del usuario

En el componente "Navbar.js":



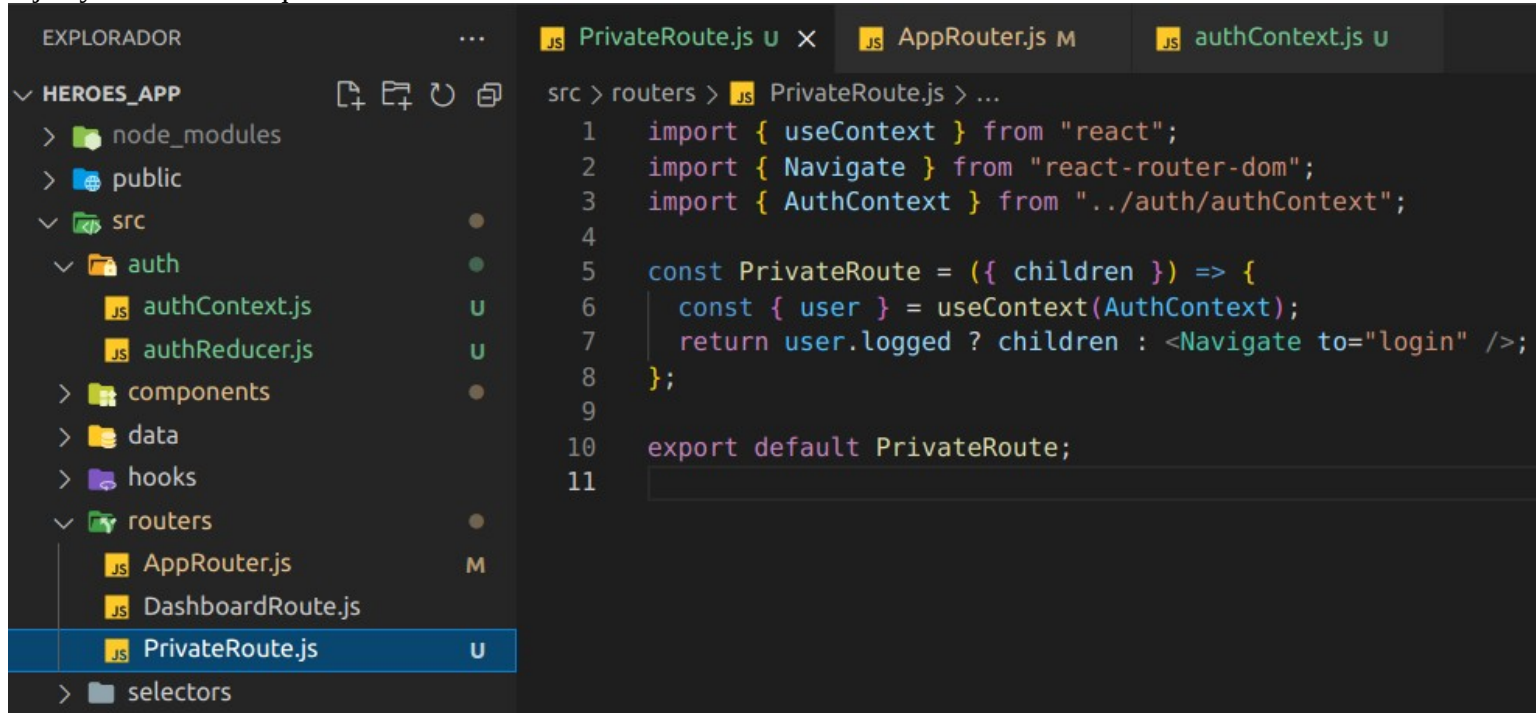
```
EXPLORADOR  ...  JS Navbar.js M  JS HeroesApp.js M  JS authReducer.js U  JS LoginScreen.js M

HEROES_APP
  node_modules
  public
  src
    auth
    components
    dc
    hero
    login
    marvel
    search
    ui
      JS Navbar.js M
    data
    hooks
    routers
    selectors
    types
      types.js
    JS HeroesApp.js M

src > components > ui > JS Navbar.js > Navbar > handleLogout
1  import { useContext } from "react";
2  import { Link, NavLink } from "react-router-dom";
3  import { useNavigate } from "react-router-dom";
4  import { AuthContext } from "../../auth/authContext";
5  import { types } from "../../types/types";
6
7  export const Navbar = () => {
8    // Leer el estado que viene de HeroesApp
9    const { user, dispatch } = useContext(AuthContext);
10    // Custom hook para la navegación
11    const navigate = useNavigate();
12
13    const handleLogout = () => {
14      // Disparar acción para cerrar sesión
15      const action = {
16        type: types.logout,
17      };
18      dispatch(action);
19
20      navigate("/login", {
21        replace: true,
22      });
23    };
24  };
25
```

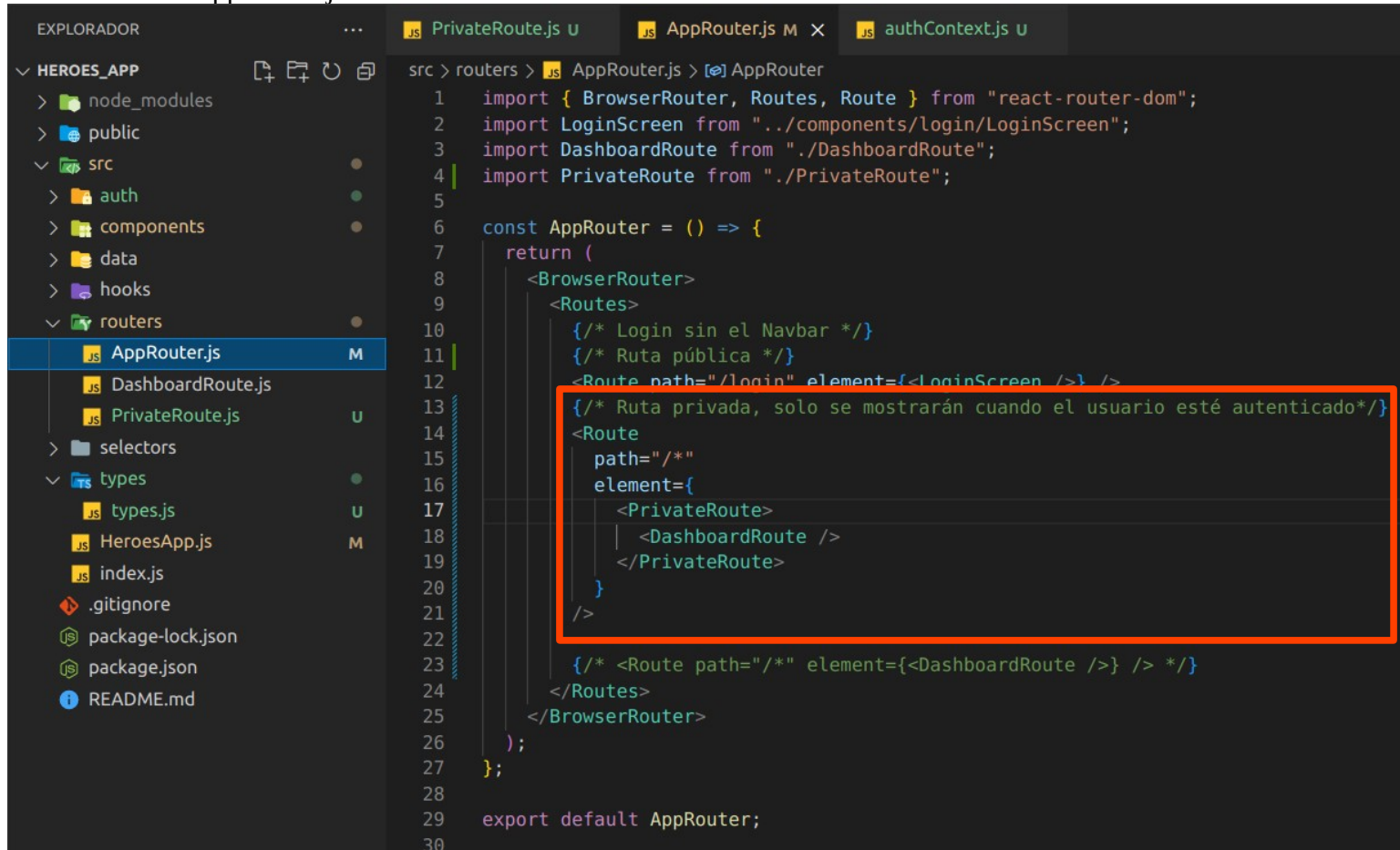
Rutas privadas

Crear componente que será renderizado solo cuando el usuario esté autenticado. Componente recibirá componentes hijos y estos son los que serán renderizados:



```
src > routers > PrivateRoute.js > ...
1  import { useContext } from "react";
2  import { Navigate } from "react-router-dom";
3  import { AuthContext } from "../auth/authContext";
4
5  const PrivateRoute = ({ children }) => {
6    const { user } = useContext(AuthContext);
7    return user.logged ? children : <Navigate to="login" />;
8  };
9
10 export default PrivateRoute;
11
```

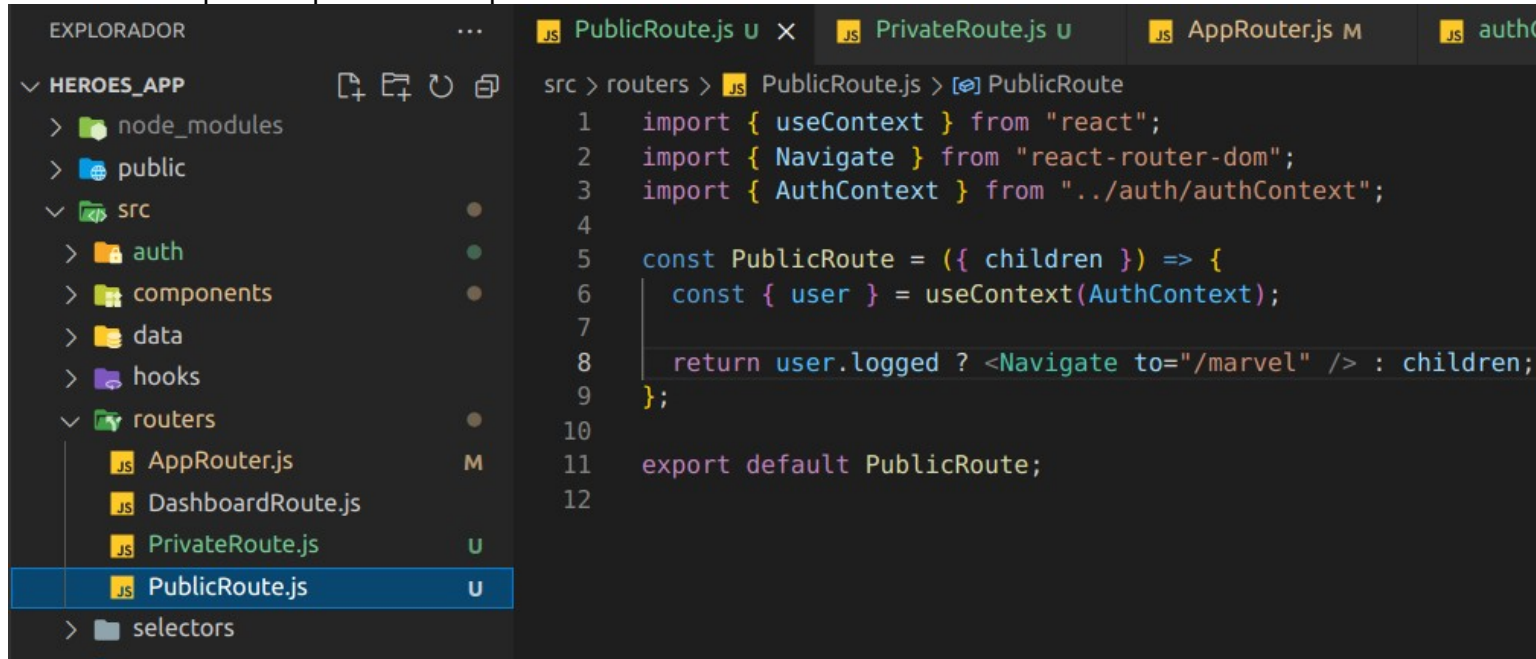
Renderizar en “AppRouter.js”



```
src > routers > AppRouter.js > [e] AppRouter
1  import { BrowserRouter, Routes, Route } from "react-router-dom";
2  import LoginScreen from "../components/login/LoginScreen";
3  import DashboardRoute from "../DashboardRoute";
4  import PrivateRoute from "../PrivateRoute";
5
6  const AppRouter = () => {
7    return (
8      <BrowserRouter>
9        <Routes>
10          { /* Login sin el Navbar */ }
11          { /* Ruta pública */ }
12          <Route path="/login" element={<LoginScreen />} />
13          { /* Ruta privada, solo se mostrarán cuando el usuario esté autenticado */ }
14          <Route
15            path="/**"
16            element={
17              <PrivateRoute>
18                <DashboardRoute />
19              </PrivateRoute>
20            }
21          />
22          { /* <Route path="/**" element={<DashboardRoute />} /> */ }
23        </Routes>
24      </BrowserRouter>
25    );
26  };
27
28 export default AppRouter;
29
30
```

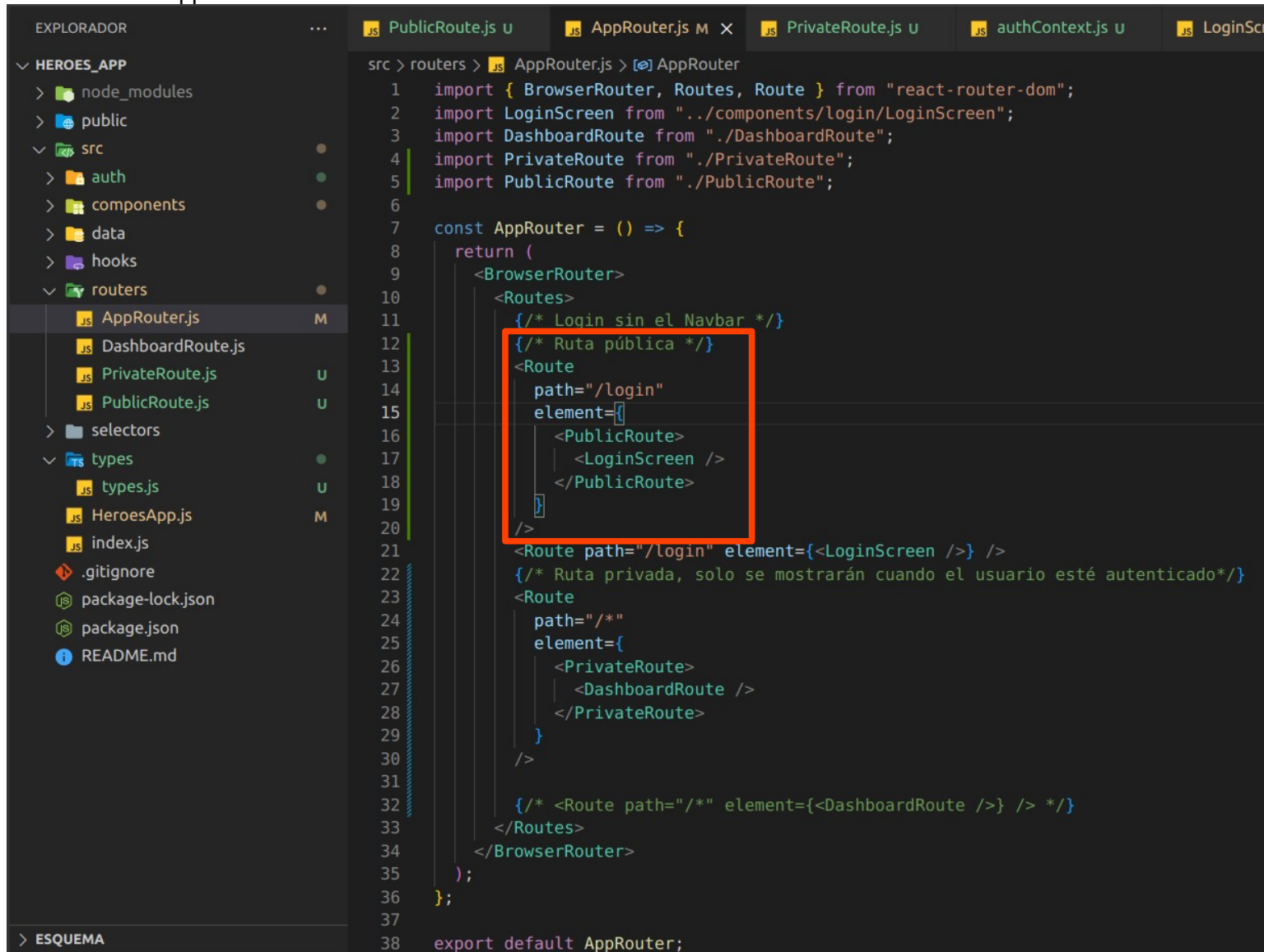

Rutas públicas

Crear otro componente para las rutas públicas:



```
EXPLORADOR src > routers > PublicRoute.js > PublicRoute
1 import { useContext } from "react";
2 import { Navigate } from "react-router-dom";
3 import { AuthContext } from "../auth/authContext";
4
5 const PublicRoute = ({ children }) => {
6   const { user } = useContext(AuthContext);
7
8   return user.logged ? <Navigate to="/marvel" /> : children;
9 };
10
11 export default PublicRoute;
```

Renderizar en AppRouter:



```
EXPLORADOR src > routers > AppRouter.js > AppRouter
1 import { BrowserRouter, Routes, Route } from "react-router-dom";
2 import LoginScreen from "../components/login/LoginScreen";
3 import DashboardRoute from "../DashboardRoute";
4 import PrivateRoute from "../PrivateRoute";
5 import PublicRoute from "../PublicRoute";
6
7 const AppRouter = () => {
8   return (
9     <BrowserRouter>
10       <Routes>
11         {/* Login sin el Navbar */}
12         {/* Ruta pública */}
13         <Route
14           path="/login"
15           element={
16             <PublicRoute>
17               <LoginScreen />
18             </PublicRoute>
19           }
20         />
21         <Route path="/login" element={<LoginScreen />} />
22         {/* Ruta privada, solo se mostrarán cuando el usuario esté autenticado*/}
23         <Route
24           path="/*"
25           element={
26             <PrivateRoute>
27               <DashboardRoute />
28             </PrivateRoute>
29           }
30         />
31         {/* <Route path="/*" element={<DashboardRoute />} /> */}
32       </Routes>
33     </BrowserRouter>
34   );
35 };
36
37 export default AppRouter;
```