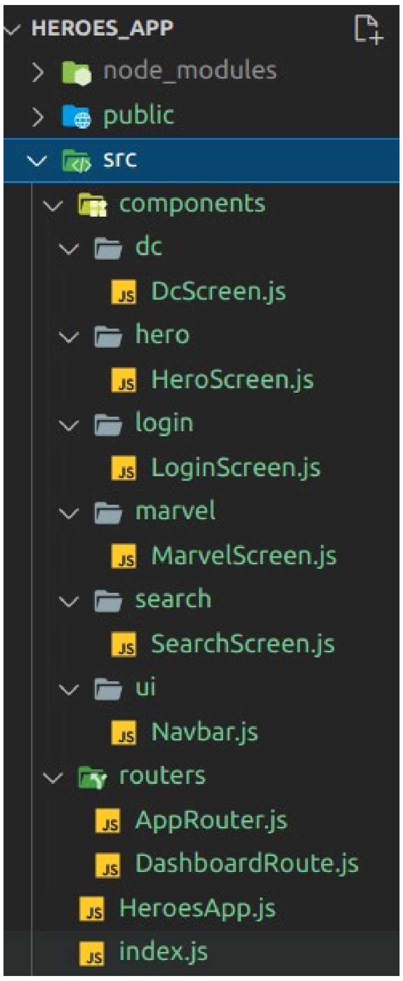
HÉROES APP

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 laas carpetas:



```
HEROES_APP
  > node_modules
  > public
  v src
    v components
      v dc
          DcScreen.js
      v hero
          HeroScreen.js
      v login
          LoginScreen.js
      v marvel
          MarvelScreen.js
      v search
          SearchScreen.js
      v ui
          Navbar.js
    v routers
        AppRouter.js
        DashboardRoute.js
      HeroesApp.js
      index.js
```

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:

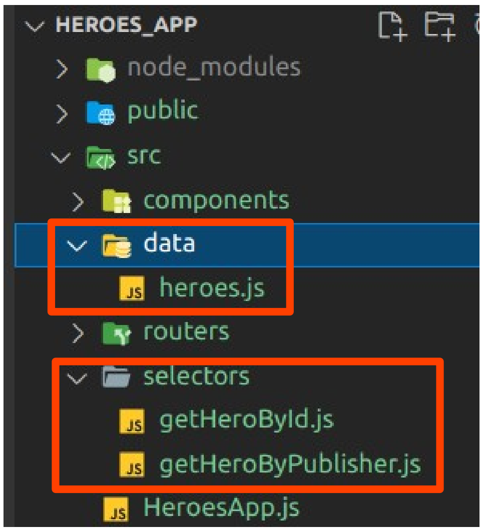
```
import { Routes, Route } from "react-router-dom";
import DcScreen from "../components/dc/DcScreen";

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:


```
HEROES_APP
  > node_modules
  > public
  v src
    > components
    v data
        heroes.js
    > routers
    v selectors
        getHeroById.js
        getHeroByPublisher.js
      HeroesApp.js
```

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:

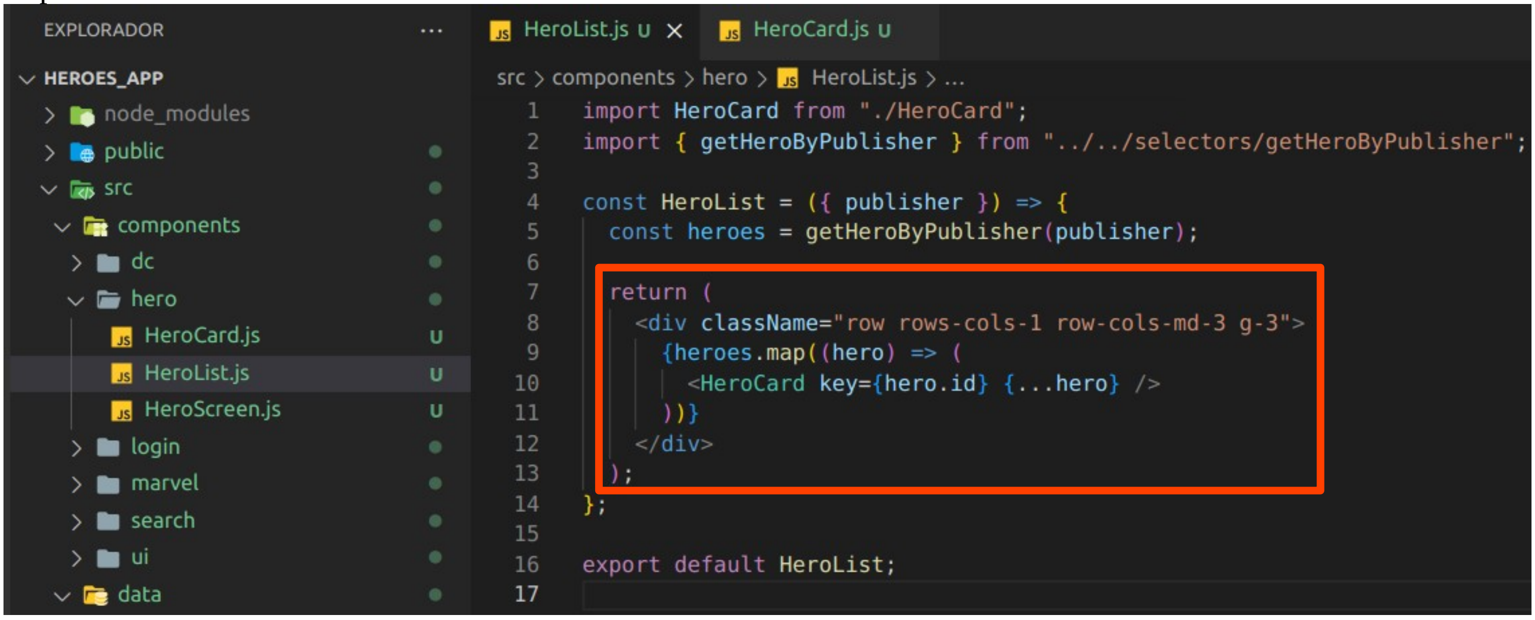
```
import HeroList from "../hero/HeroList";

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":

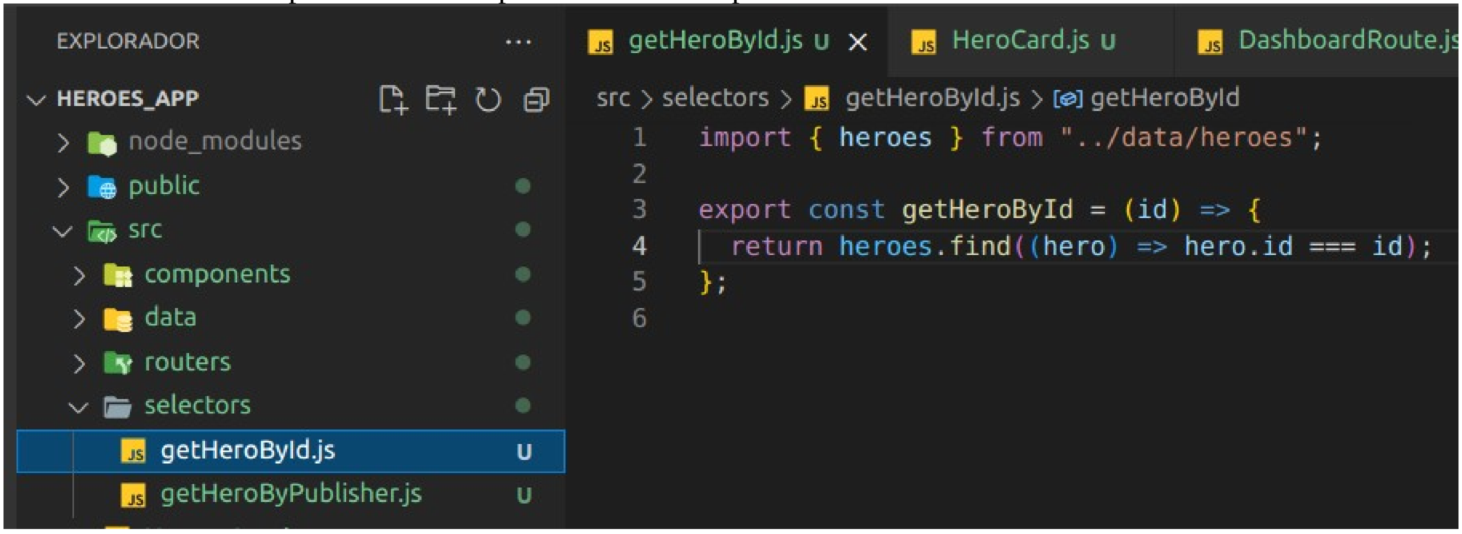
```
import { Link } from "react-router-dom";

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":



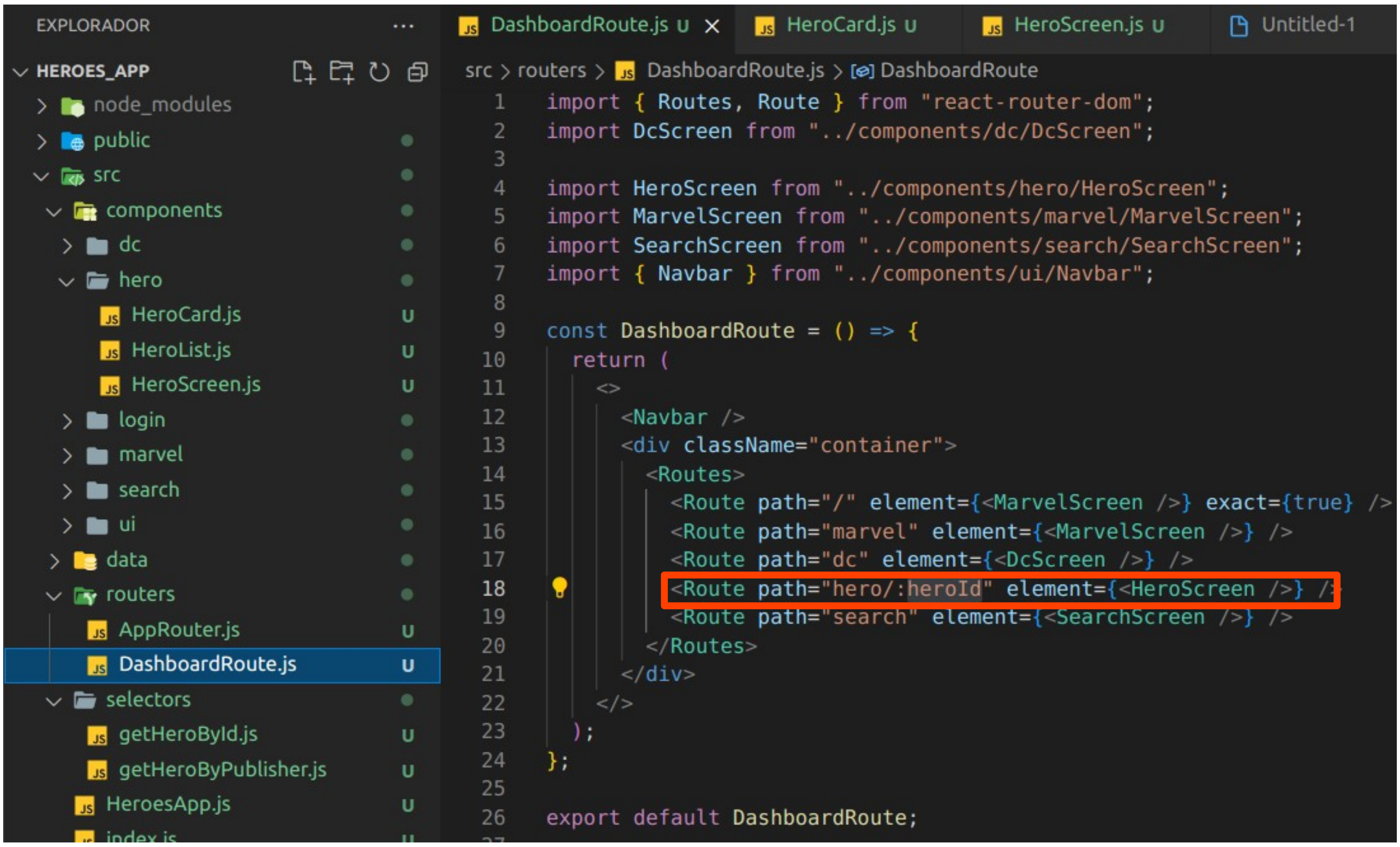
```
import HeroCard from "./HeroCard";
import { getHeroByPublisher } from "../../selectors/getHeroByPublisher";

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

  return (
    <div className="row rows-cols-1 row-cols-md-3 g-3">
      {heroes.map((hero) => (
        <HeroCard key={hero.id} {...hero} />
      ))}
    </div>
  );
};

export default HeroList;
```

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



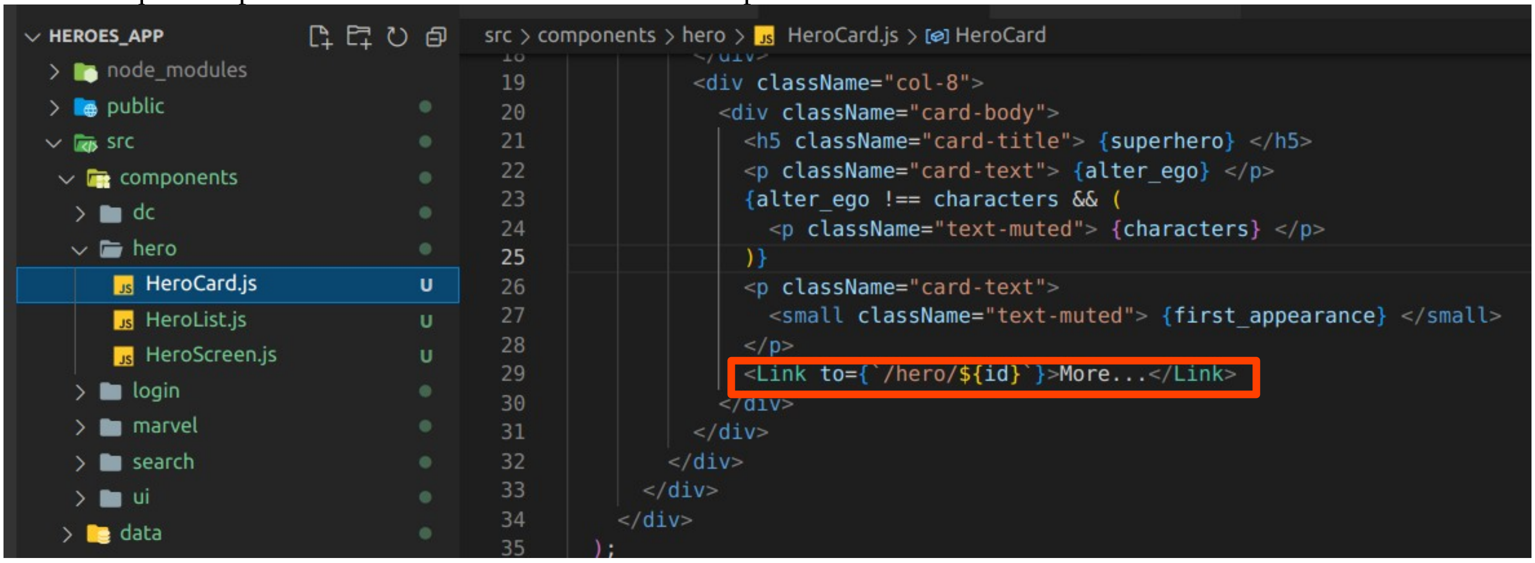
```
import { heroes } from "../data/heroes";

export const getHeroById = (id) => {
  return heroes.find((hero) => hero.id === id);
};
```

Crear la ruta en “DashboadRoute”:



```
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 />
      <div className="container">
        <Routes>
          <Route path="/" element={<MarvelScreen />} exact={true} />
          <Route path="marvel" element={<MarvelScreen />} />
          <Route path="dc" element={<DcScreen />} />
          <Route path="hero/:heroId" element={<HeroScreen />} />
          <Route path="search" element={<SearchScreen />} />
        </Routes>
      </div>
    </>
  );
};

export default DashboardRoute;
```

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



```
HEROES_APP
  node_modules
  public
  src
    components
      dc
      hero
        HeroCard.js
        HeroList.js
        HeroScreen.js
      login
      marvel
      search
      ui
    data
```

src > components > hero > HeroCard.js > HeroCard

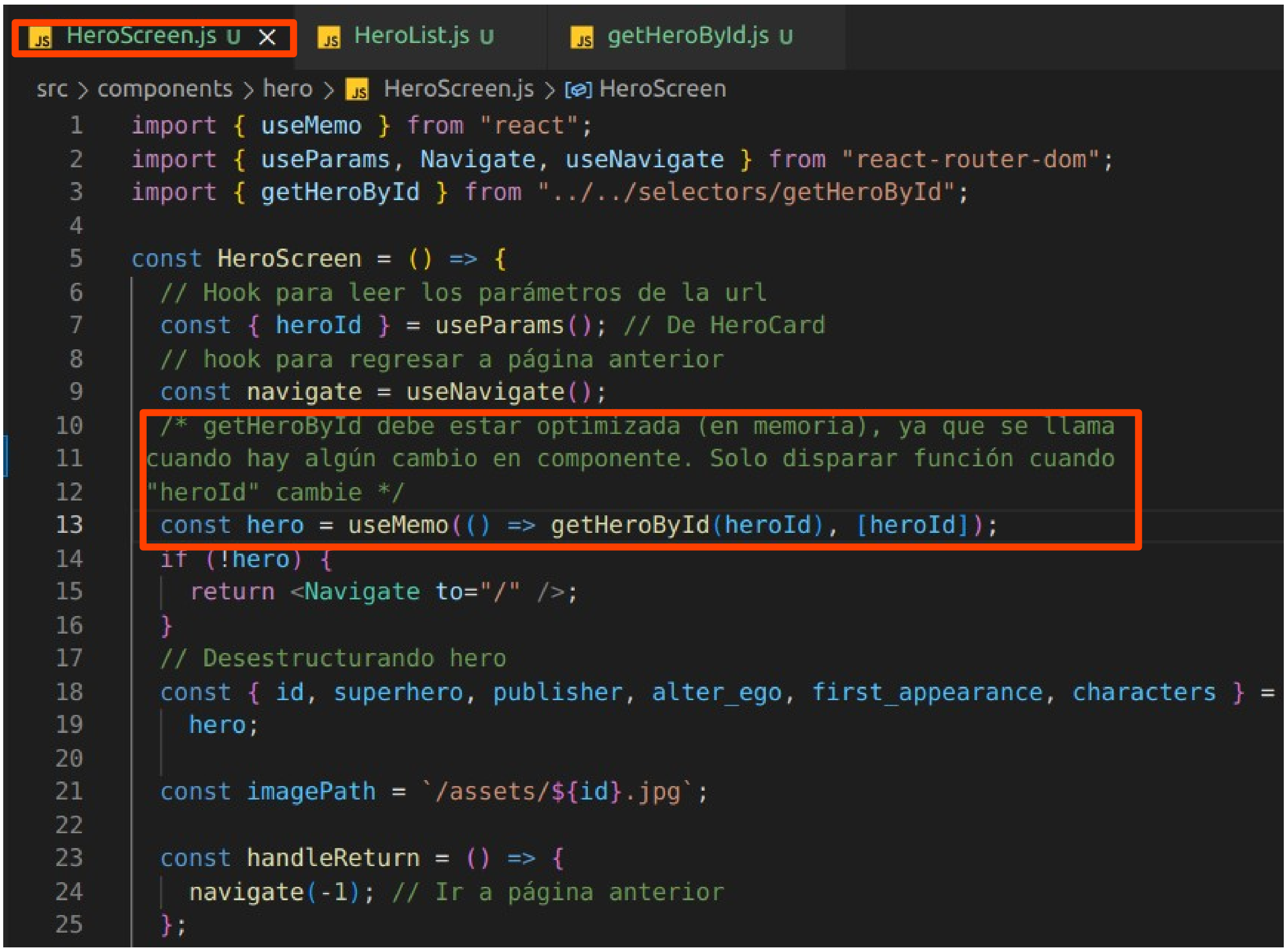
```
          <div className="col-8">
            <div className="card-body">
              <h5 className="card-title"> {superhero} </h5>
              <p className="card-text"> {alter_ego} </p>
              {alter_ego !== characters && (
                <p className="text-muted"> {characters} </p>
              )}
              <p className="card-text">
                <small className="text-muted"> {first_appearance} </small>
              </p>
              <Link to={`/hero/${id}`}>More...</Link>
            </div>
          </div>
        </div>
      </div>
    );
```

$$$

Tomar el id en el componente HeroScreen:

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

const HeroScreen = () => {
// Hook para leer los parámetros de la url
const { heroId } = useParams(); // De HeroCard
// hook para regresar a pagina anterior
const navigate = useNavigate();
const hero = getHeroById(heroId);
if (!hero) {
return <Navigate to="/" />;
}
// Desestructurando hero
const { id, superhero, publisher, alter_ego, first_appearance, characters } =
hero;

const imagePath = `/assets/${id}.jpg`;

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;
```

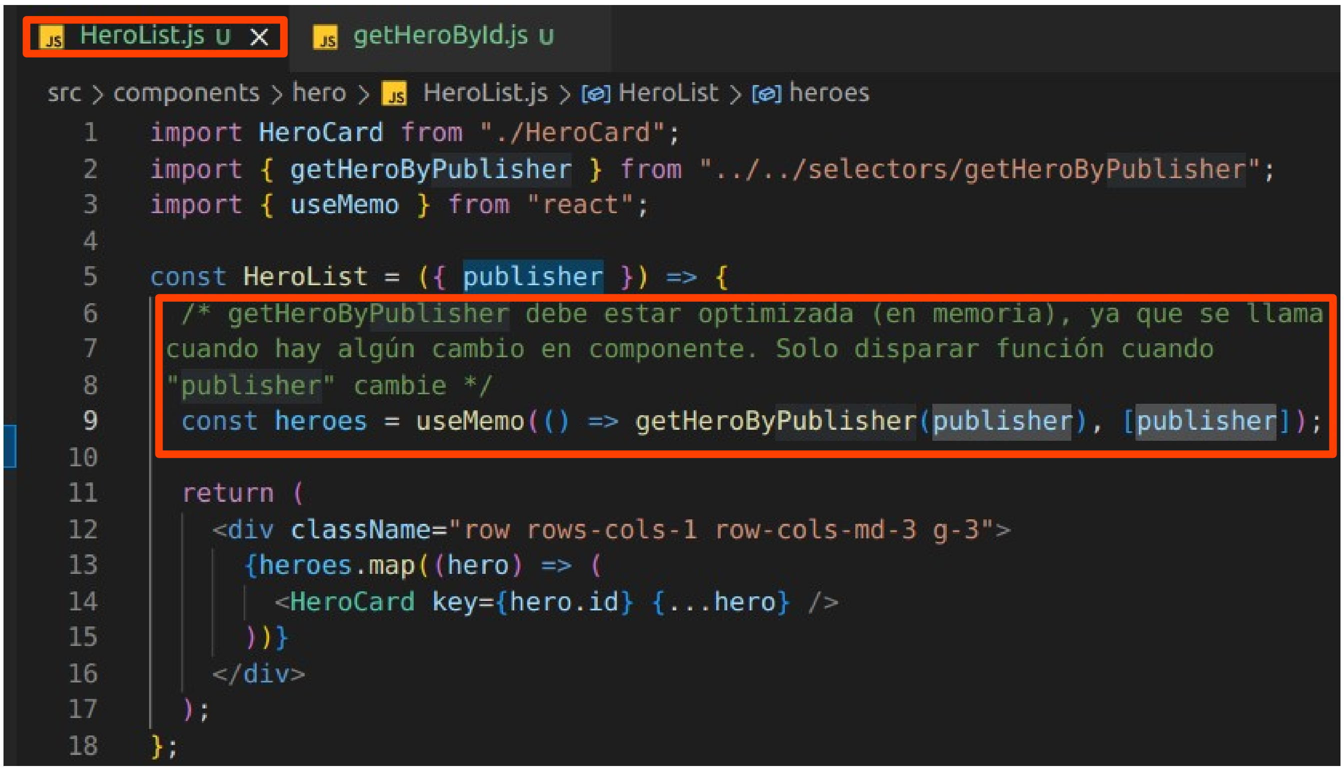
Nota useMemo
Usar hook “useMemo” para optimizar funciones “getHeroById” y “getHeroByPublisher”:



HeroScreen.js U × | HeroList.js U | getHeroById.js U

src > components > hero > HeroScreen.js > HeroScreen

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

const HeroScreen = () => {
  // Hook para leer los parámetros de la url
  const { heroId } = useParams(); // De HeroCard
  // hook para regresar a página anterior
  const navigate = useNavigate();
  /* getHeroById debe estar optimizada (en memoria), ya que se llama
cuando hay algún cambio en componente. Solo disparar función cuando
"heroId" cambie */
  const hero = useMemo(() => getHeroById(heroId), [heroId]);
  if (!hero) {
    return <Navigate to="/" />;
  }
  // Desestructurando hero
  const { id, superhero, publisher, alter_ego, first_appearance, characters } =
    hero;

  const imagePath = `/assets/${id}.jpg`;

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

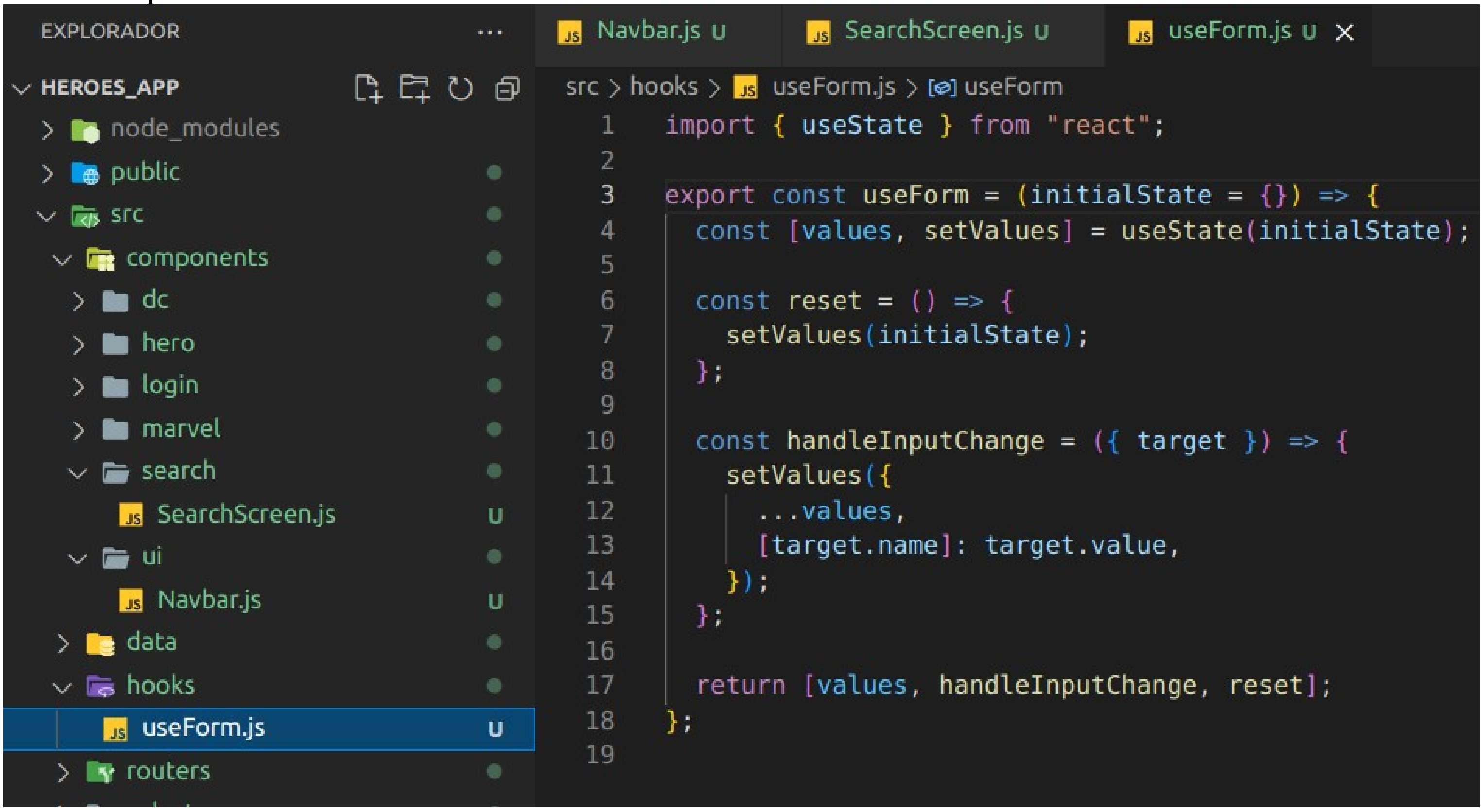


HeroList.js U × | getHeroById.js U

src > components > hero > HeroList.js > HeroList > heroes

```
import HeroCard from "./HeroCard";
import { getHeroByPublisher } from "../../selectors/getHeroByPublisher";
import { useMemo } from "react";

const HeroList = ({ publisher }) => {
  /* getHeroByPublisher debe estar optimizada (en memoria), ya que se llama
cuando hay algún cambio en componente. Solo disparar función cuando
"publisher" cambie */
  const heroes = useMemo(() => getHeroByPublisher(publisher), [publisher]);

  return (
    <div className="row rows-cols-1 row-cols-md-3 g-3">
      {heroes.map((hero) => (
        <HeroCard key={hero.id} {...hero} />
      ))}
    </div>
  );
};
```

SearchComponent
Crear hook para formularios:

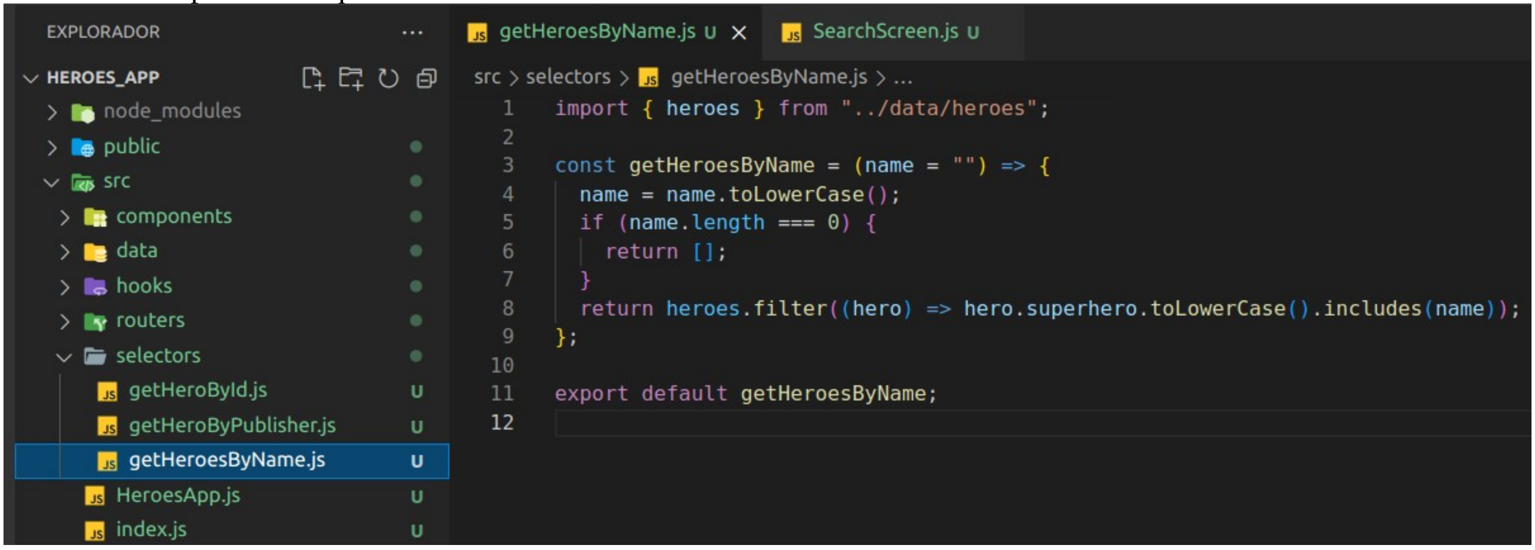


EXPLORADOR
HEROES_APP
node_modules
public
src
components
dc
hero
login
marvel
search
SearchScreen.js
ui
Navbar.js
data
hooks
useForm.js
routers

src > hooks > useForm.js > useForm

```javascript
import { useState } from "react";

export const useForm = (initialState = {}) => {
  const [values, setValues] = useState(initialState);

  const reset = () => {
    setValues(initialState);
  };

  const handleInputChange = ({ target }) => {
    setValues({
      ...values,
      [target.name]: target.value,
    });
  };

  return [values, handleInputChange, reset];
};
```

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 > ...

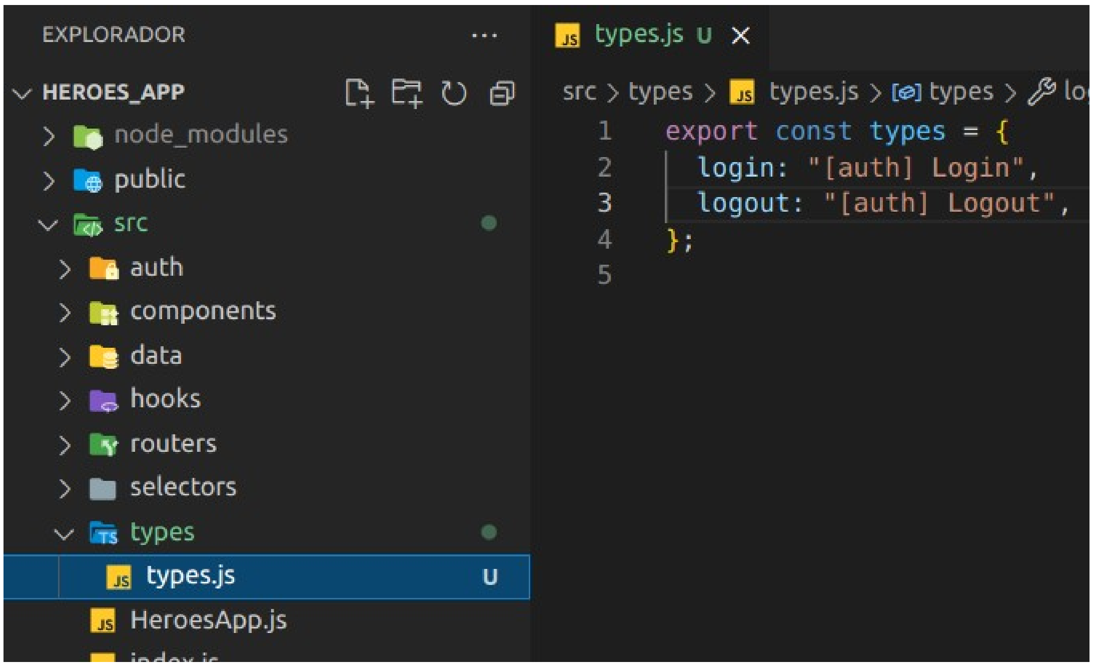
```javascript
import { heroes } from "../data/heroes";

const getHeroesByName = (name = "") => {
  name = name.toLowerCase();
  if (name.length === 0) {
    return [];
  }
  return heroes.filter((hero) => hero.superhero.toLowerCase().includes(name));
};

export default getHeroesByName;
```

En “SerachScreen.js”:
Usar el paquete query-string para trabajar con los parámetros que vienen en la ruta:
https://www.npmjs.com/package/query-string

```javascript
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">
Seacrh...
</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>
</div>
);
};

export default SearchScreen;
```

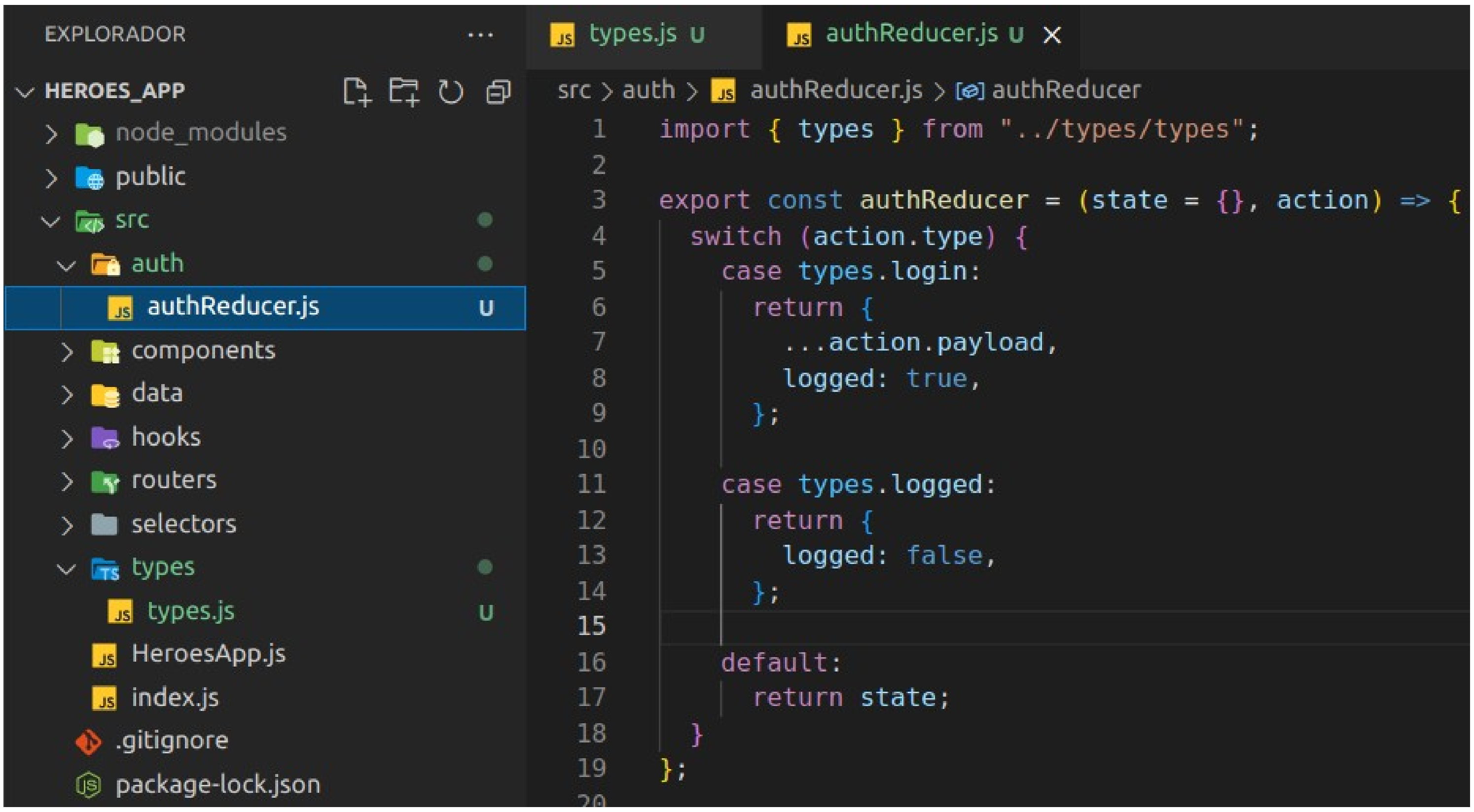
Protección de rutas

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)



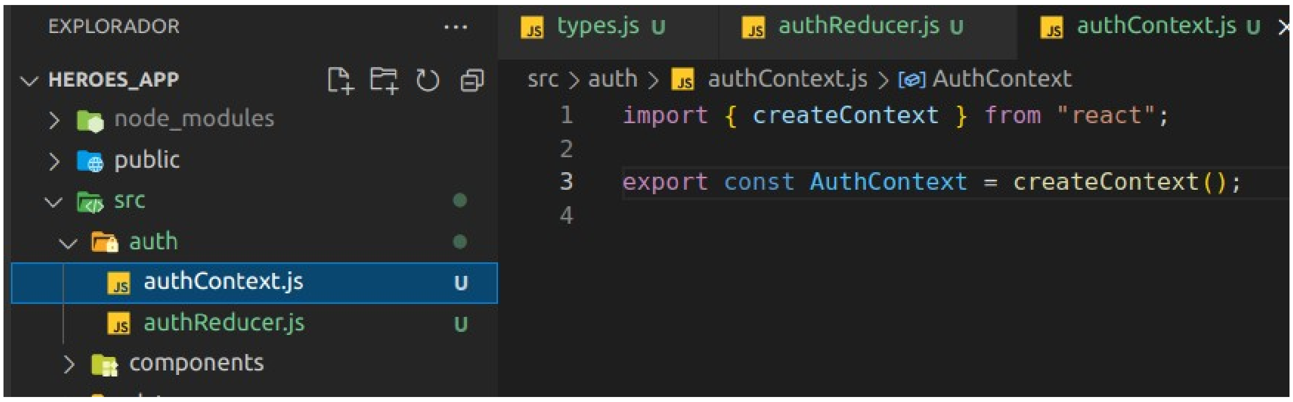
```js
export const types = {
  login: "[auth] Login",
  logout: "[auth] Logout",
};
```

Crear un reducer en “auth”:



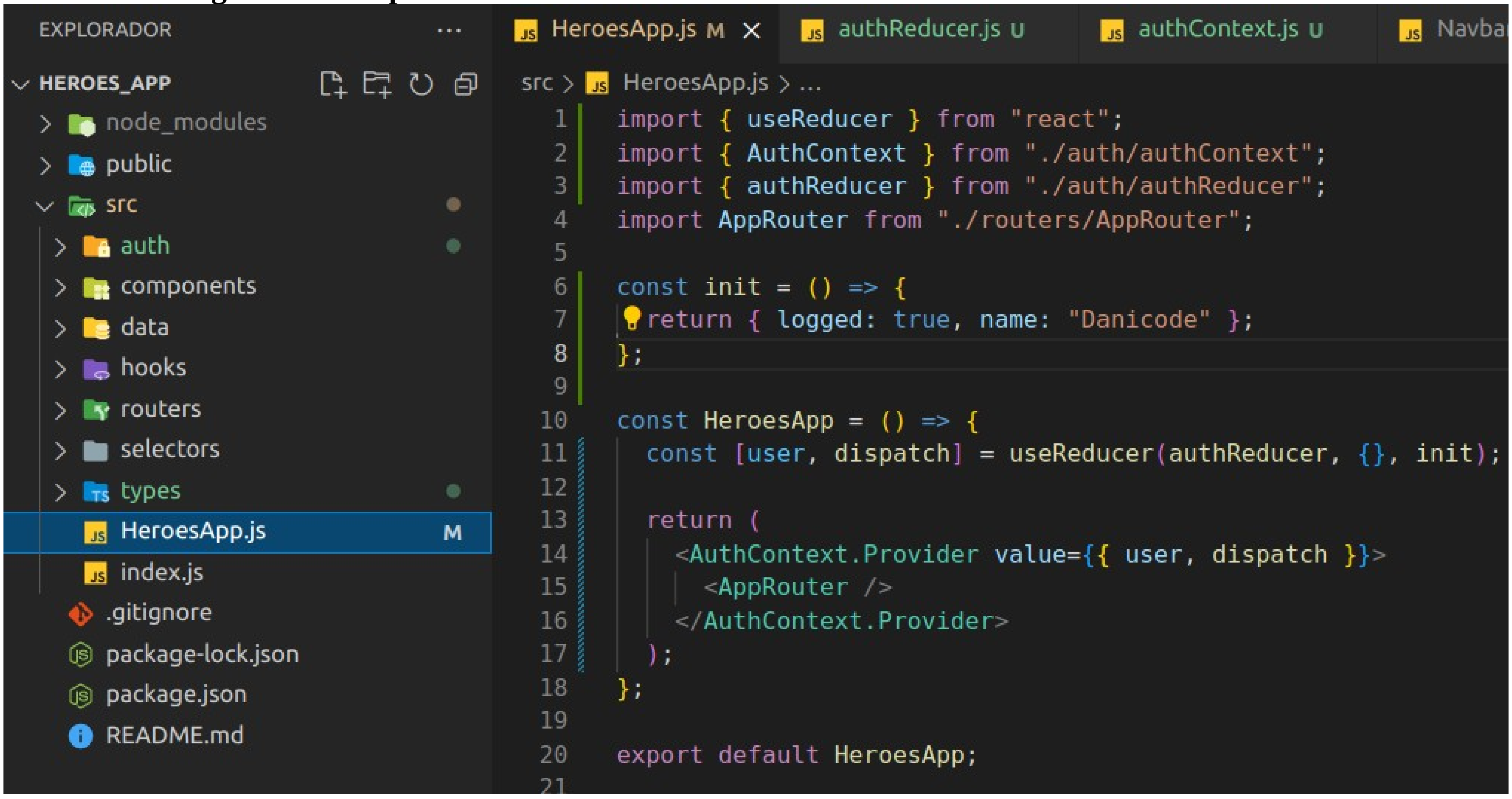
```js
import { types } from "../types/types";

export const authReducer = (state = {}, action) => {
  switch (action.type) {
    case types.login:
      return {
        ...action.payload,
        logged: true,
      };

    case types.logged:
      return {
        logged: false,
      };

    default:
      return state;
  }
};
```

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).



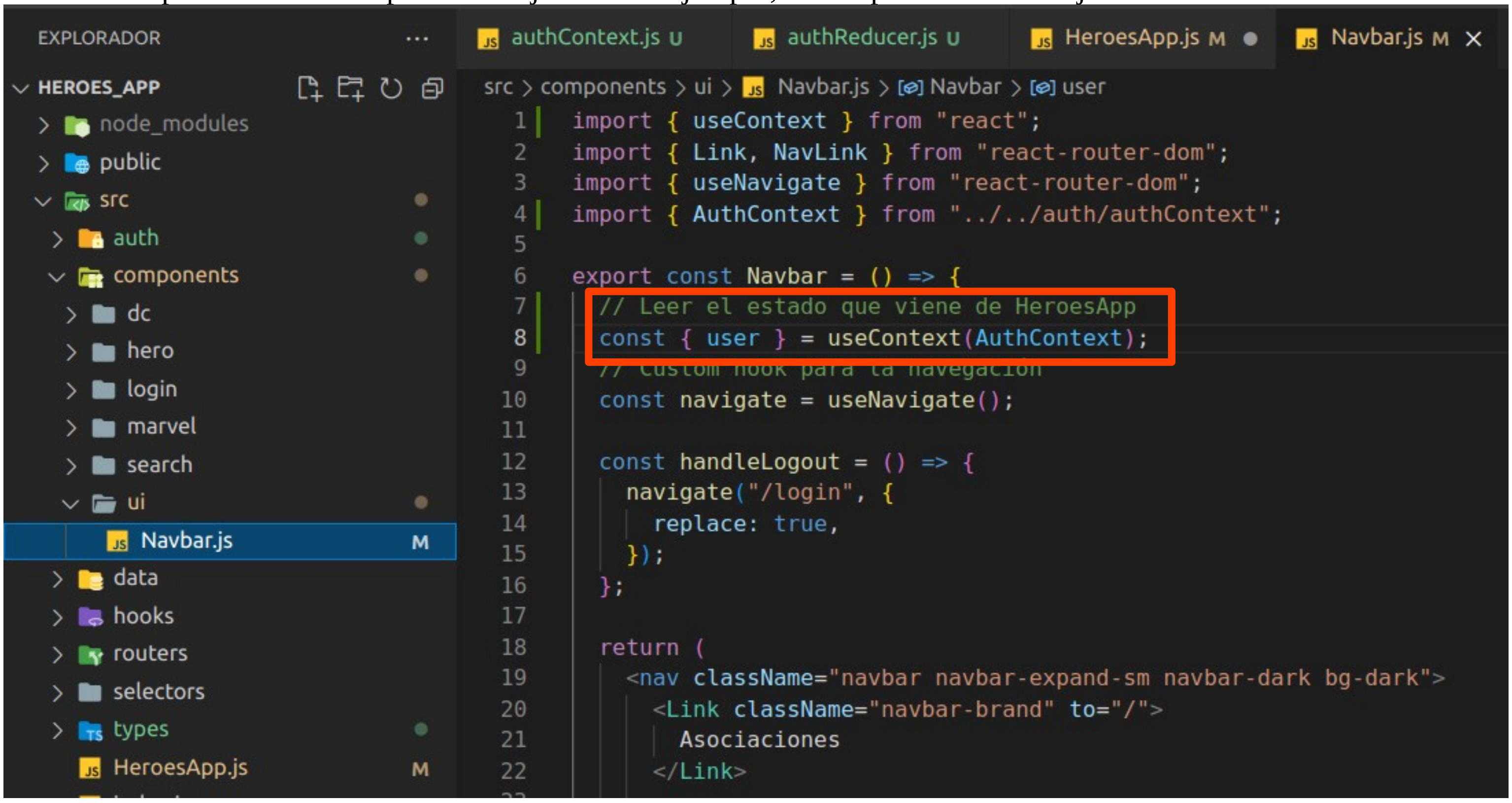
```
import { createContext } from "react";

export const AuthContext = createContext();
```

Proveer estado global de la aplicación



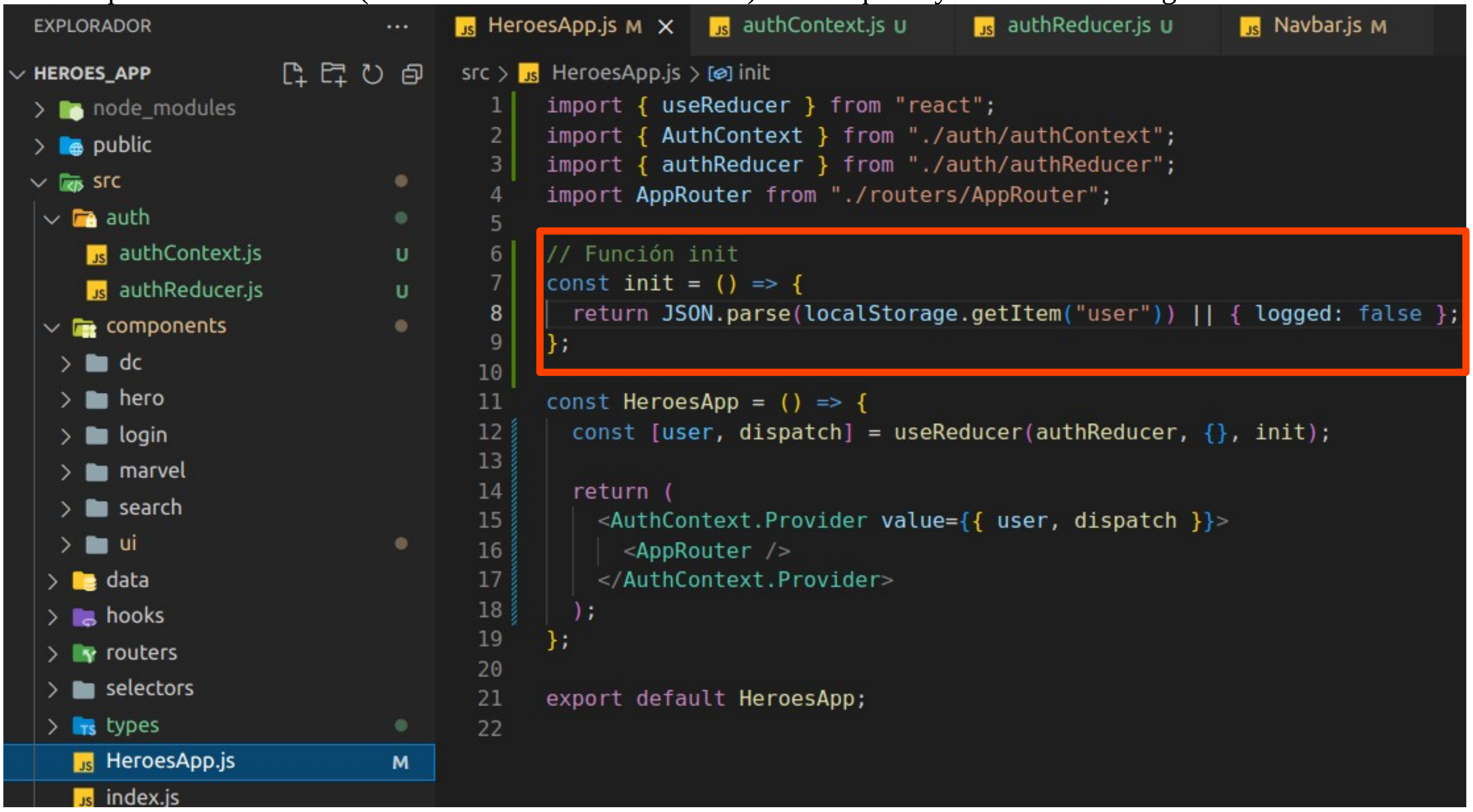
```
import { useReducer } from "react";
import { AuthContext } from "./auth/authContext";
import { authReducer } from "./auth/authReducer";
import AppRouter from "./routers/AppRouter";

const init = () => {
  return { logged: true, name: "Danicode" };
};

const HeroesApp = () => {
  const [user, dispatch] = useReducer(authReducer, {}, init);

  return (
    <AuthContext.Provider value={{ user, dispatch }}>
      <AppRouter />
    </AuthContext.Provider>
  );
};

export default HeroesApp;
```

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”:



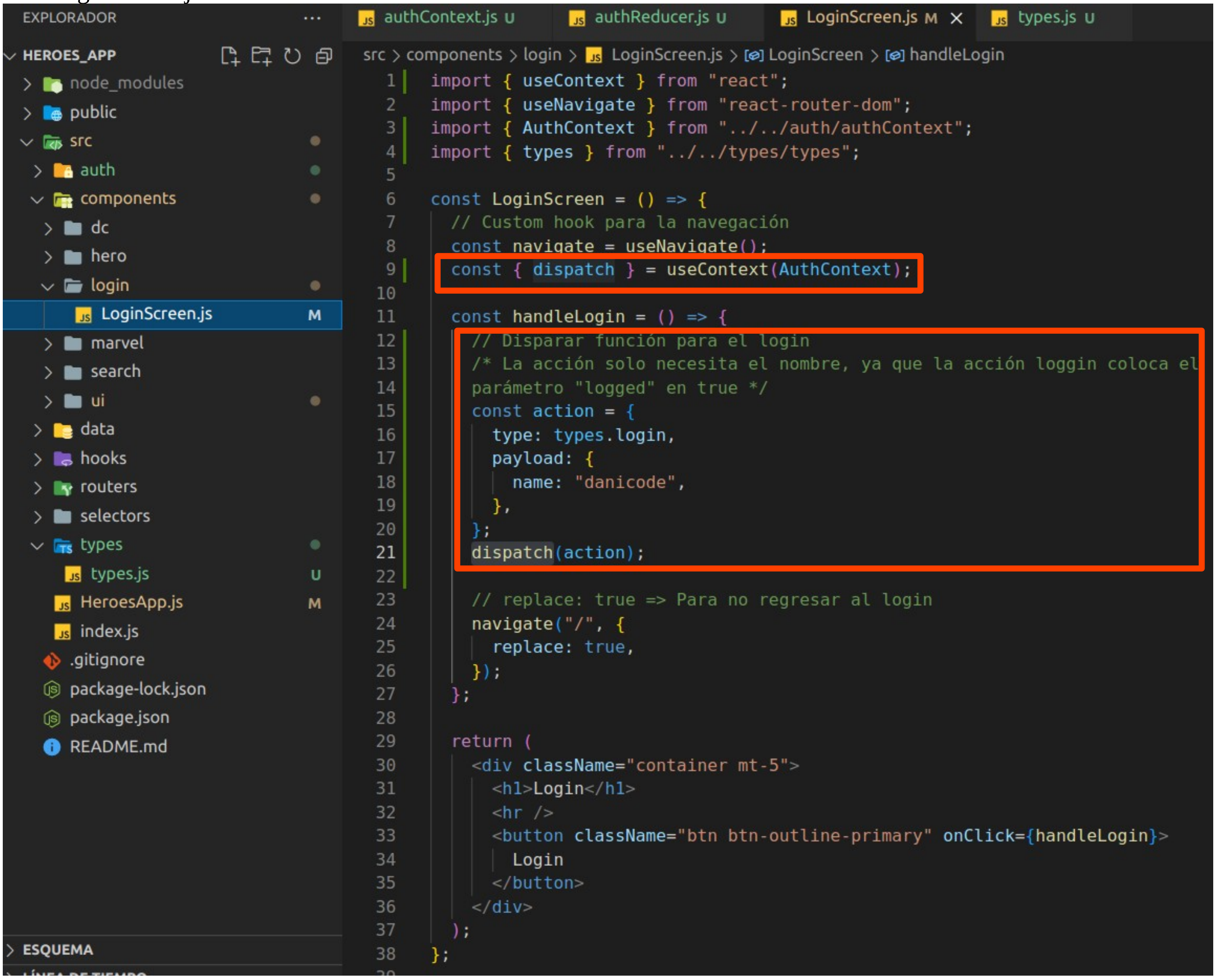
```javascript
import { useContext } from "react";
import { Link, NavLink } from "react-router-dom";
import { useNavigate } from "react-router-dom";
import { AuthContext } from "../../auth/authContext";

export const Navbar = () => {
  // Leer el estado que viene de HeroesApp
  const { user } = useContext(AuthContext);
  // Custom hook para la navegación
  const navigate = useNavigate();

  const handleLogout = () => {
    navigate("/login", {
      replace: true,
    });
  };

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

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



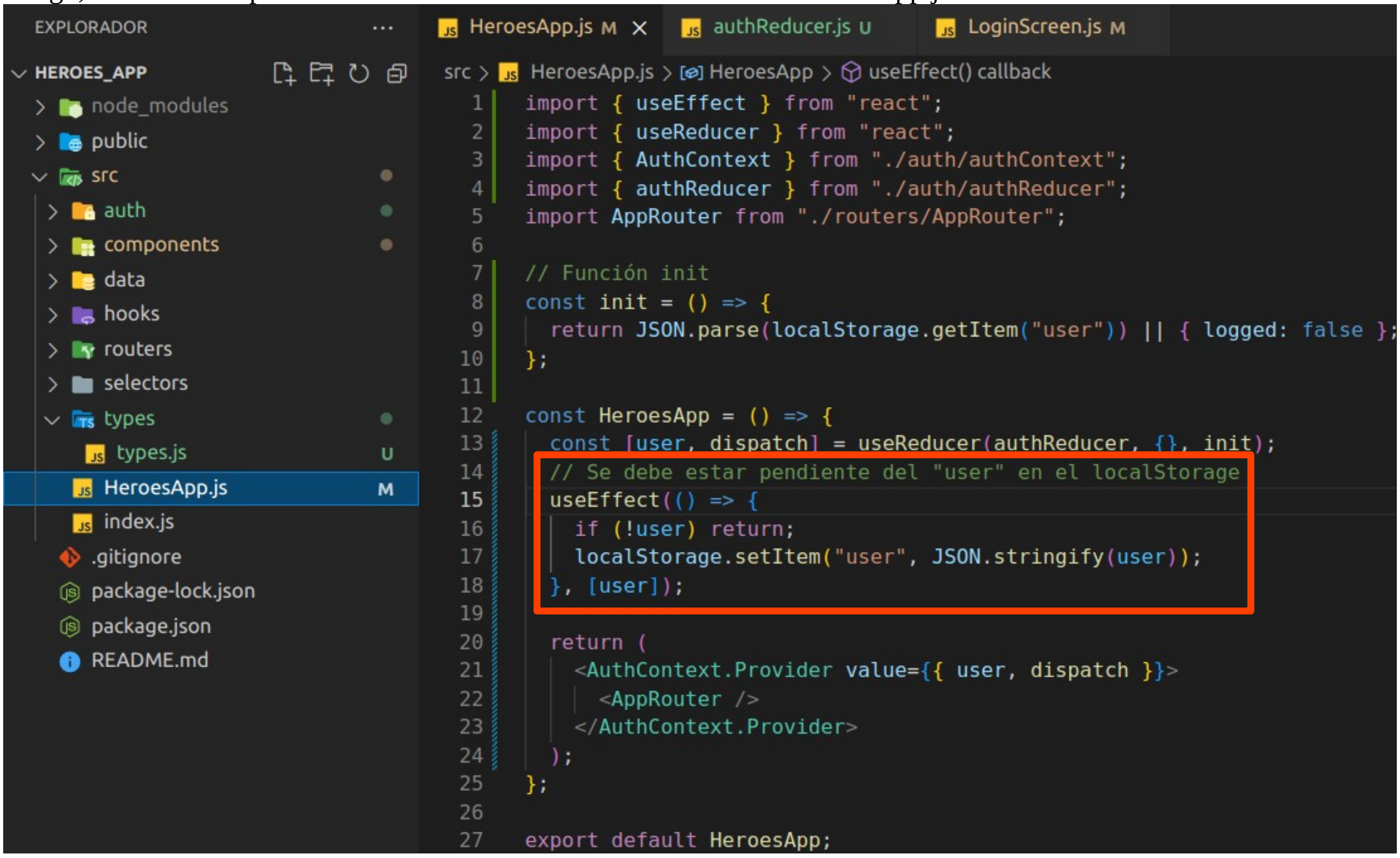
```javascript
import { useReducer } from "react";
import { AuthContext } from "./auth/authContext";
import { authReducer } from "./auth/authReducer";
import AppRouter from "./routers/AppRouter";

// Función init
const init = () => {
  return JSON.parse(localStorage.getItem("user")) || { logged: false };
};

const HeroesApp = () => {
  const [user, dispatch] = useReducer(authReducer, {}, init);

  return (
    <AuthContext.Provider value={{ user, dispatch }}>
      <AppRouter />
    </AuthContext.Provider>
  );
};

export default HeroesApp;
```

Login de un usuario
En “LoginScreen.js”:



```javascript
import { useContext } from "react";
import { useNavigate } from "react-router-dom";
import { AuthContext } from "../../auth/authContext";
import { types } from "../../types/types";

const LoginScreen = () => {
  // Custom hook para la navegación
  const navigate = useNavigate();
  const { dispatch } = useContext(AuthContext);

  const handleLogin = () => {
    // Disparar función para el login
    /* La acción solo necesita el nombre, ya que la acción loggin coloca el
    parámetro "logged" en true */
    const action = {
      type: types.login,
      payload: {
        name: "danicode",
      },
    };
    dispatch(action);

    // 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>
  );
};
```

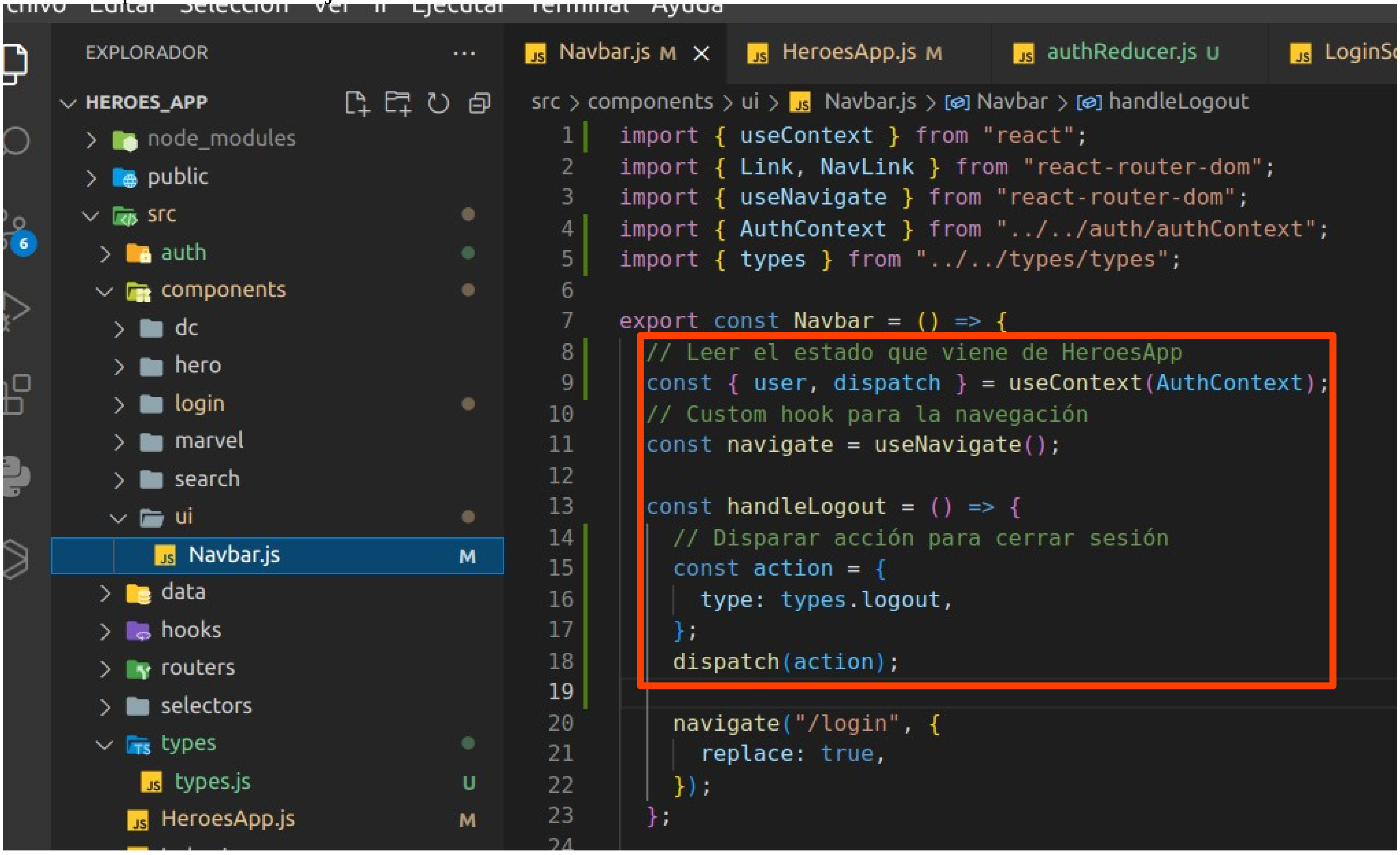
Luego, se debe estar pendiente al cambio de estado del usuario. En HeroesApp.js:



```
import { useEffect } from "react";
import { useReducer } from "react";
import { AuthContext } from "./auth/authContext";
import { authReducer } from "./auth/authReducer";
import AppRouter from "./routers/AppRouter";

// Función init
const init = () => {
  return JSON.parse(localStorage.getItem("user")) || { logged: false };
};

const HeroesApp = () => {
  const [user, dispatch] = useReducer(authReducer, {}, init);
  // Se debe estar pendiente del "user" en el localStorage
  useEffect(() => {
    if (!user) return;
    localStorage.setItem("user", JSON.stringify(user));
  }, [user]);

  return (
    <AuthContext.Provider value={{ user, dispatch }}>
      <AppRouter />
    </AuthContext.Provider>
  );
};

export default HeroesApp;
```

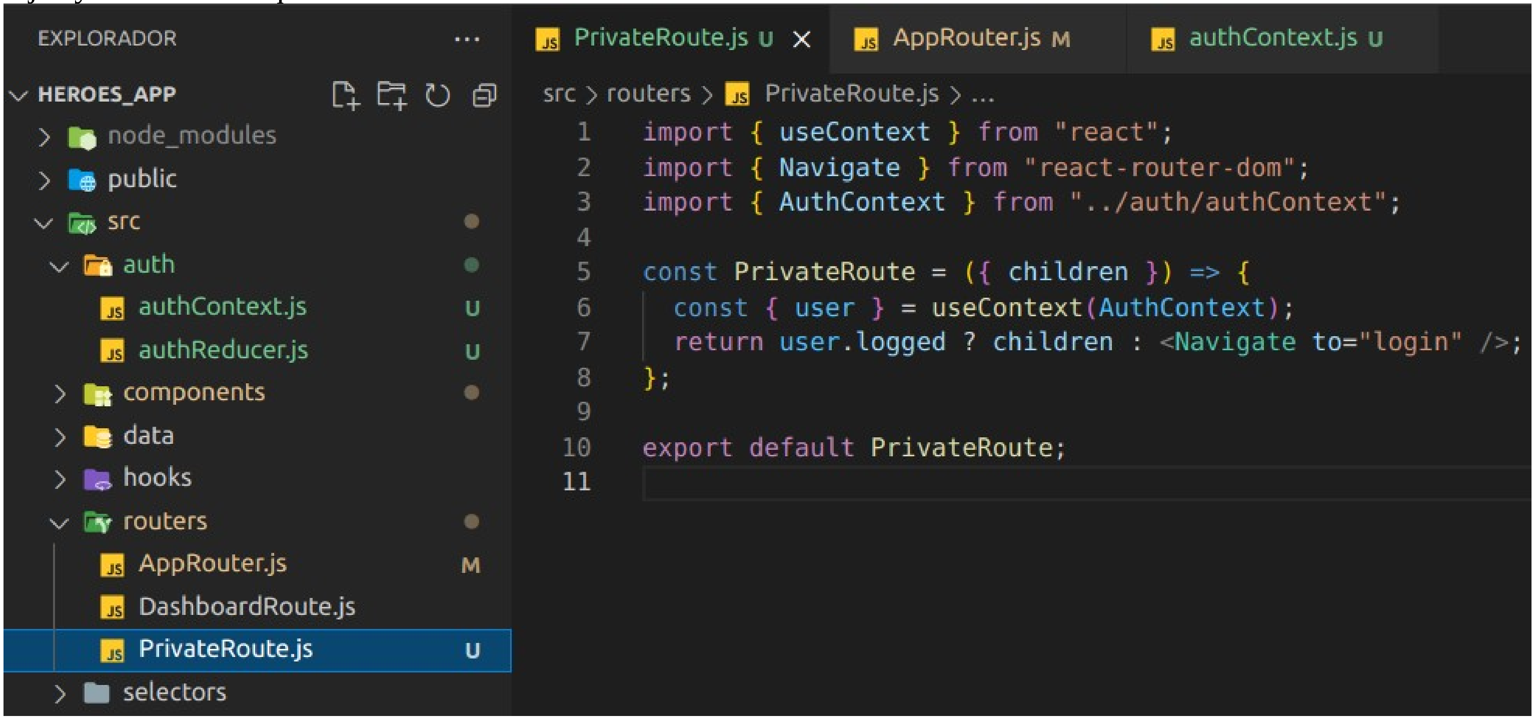
Logout del usuario

En el componente “Navbar.js”:



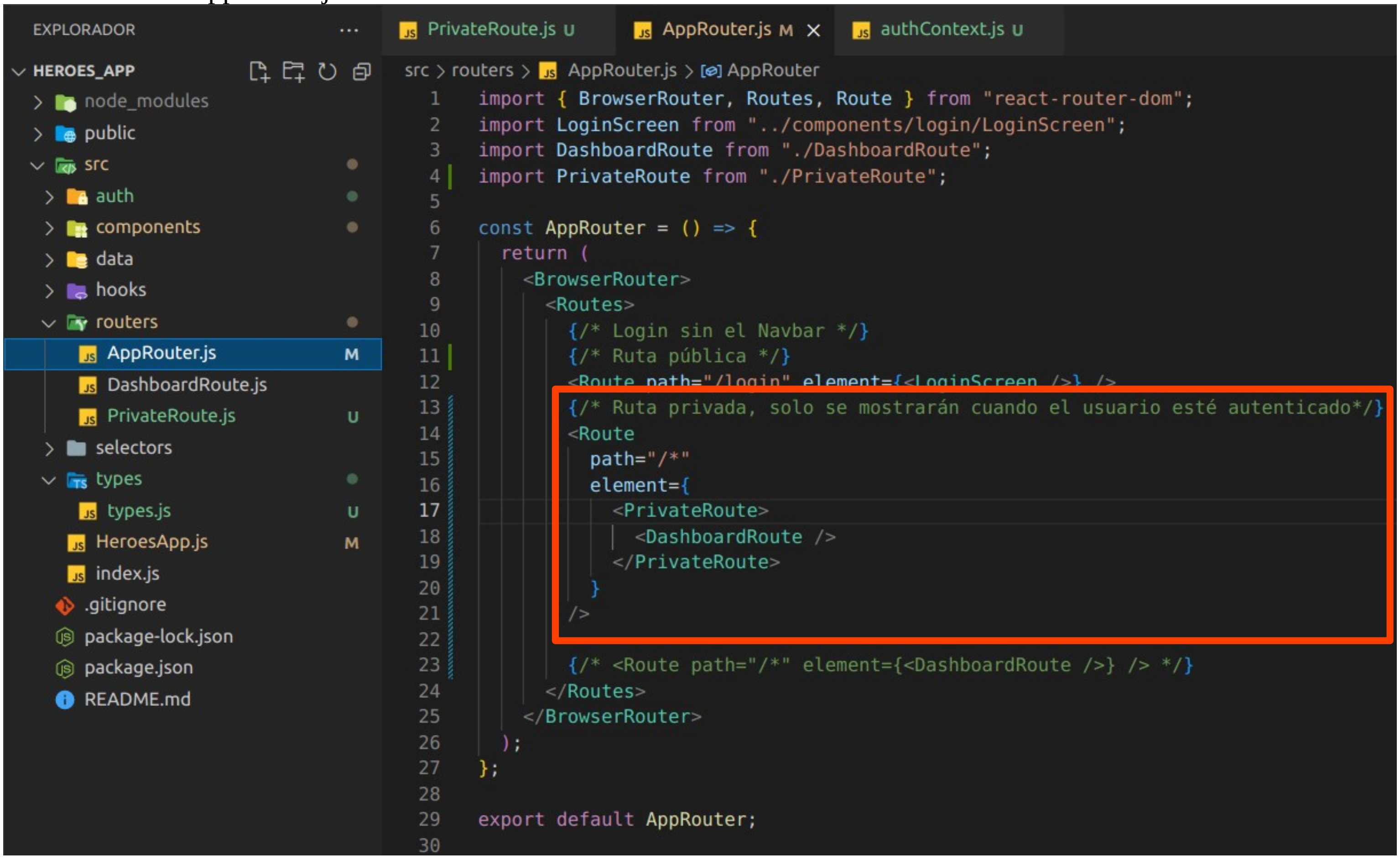
```
import { useContext } from "react";
import { Link, NavLink } from "react-router-dom";
import { useNavigate } from "react-router-dom";
import { AuthContext } from "../../auth/authContext";
import { types } from "../../types/types";

export const Navbar = () => {
  // Leer el estado que viene de HeroesApp
  const { user, dispatch } = useContext(AuthContext);
  // Custom hook para la navegación
  const navigate = useNavigate();

  const handleLogout = () => {
    // Disparar acción para cerrar sesión
    const action = {
      type: types.logout,
    };
    dispatch(action);

    navigate("/login", {
      replace: true,
    });
  };
```

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



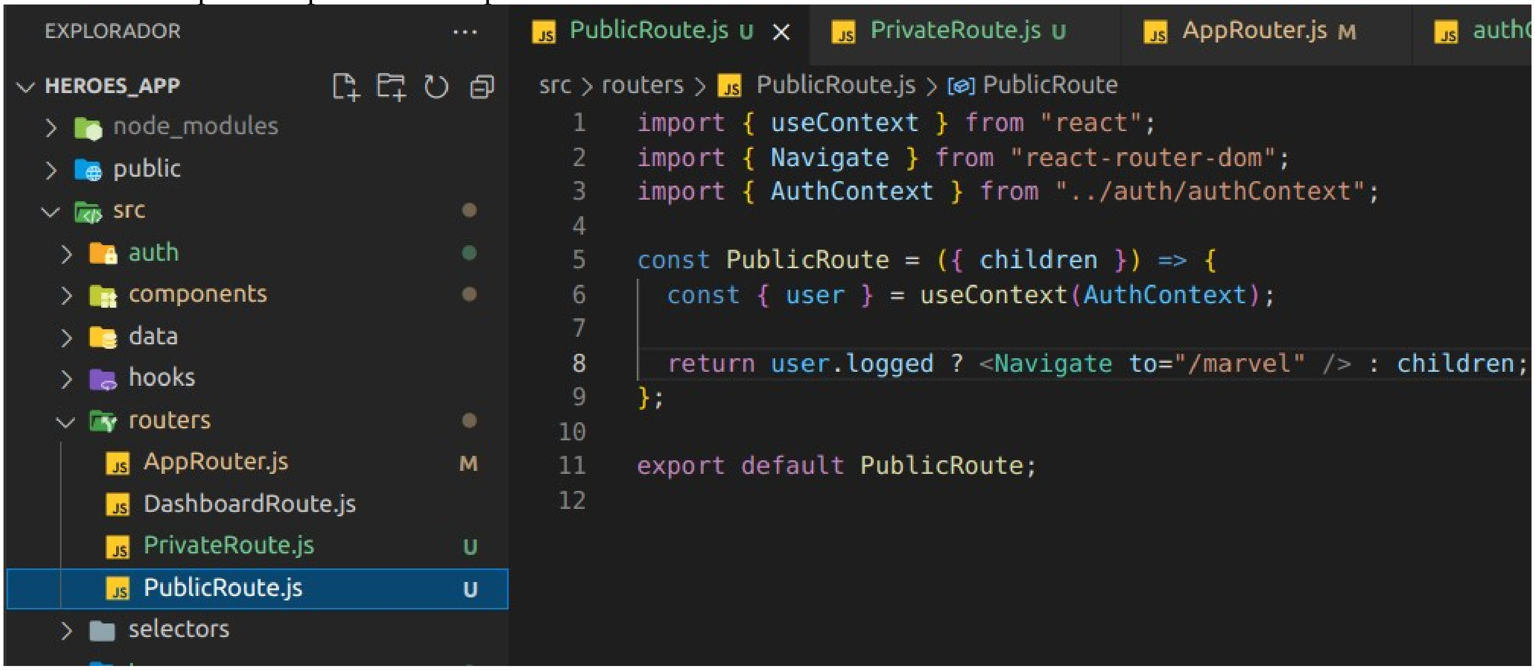
```
import { useContext } from "react";
import { Navigate } from "react-router-dom";
import { AuthContext } from "../auth/authContext";

const PrivateRoute = ({ children }) => {
  const { user } = useContext(AuthContext);
  return user.logged ? children : <Navigate to="login" />;
};

export default PrivateRoute;
```

Renderizar en "AppRouter.js"



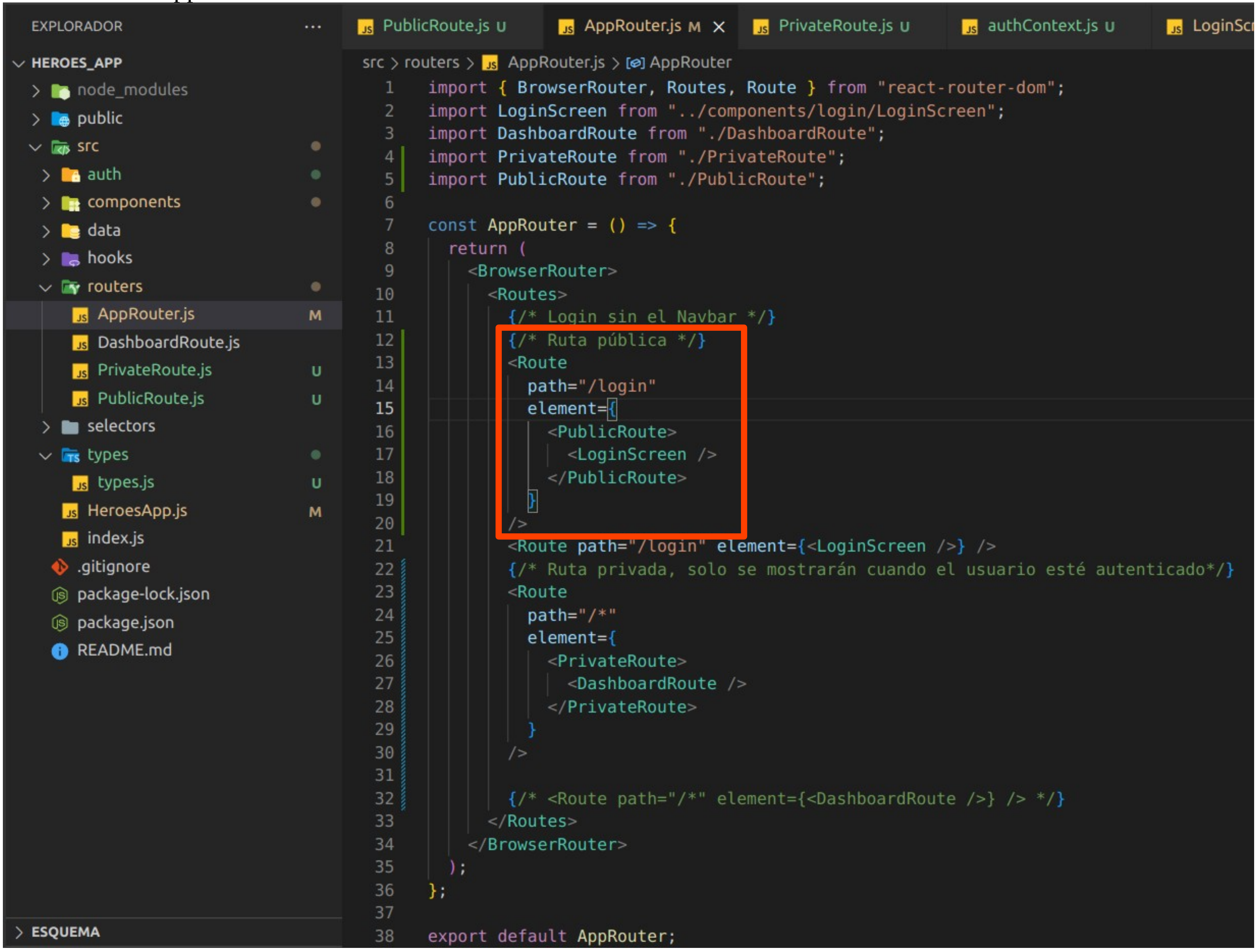
```
import { BrowserRouter, Routes, Route } from "react-router-dom";
import LoginScreen from "../components/login/LoginScreen";
import DashboardRoute from "./DashboardRoute";
import PrivateRoute from "./PrivateRoute";

const AppRouter = () => {
  return (
    <BrowserRouter>
      <Routes>
        {/* Login sin el Navbar */}
        {/* Ruta pública */}
        <Route path="/login" element={<LoginScreen />} />
        {/* Ruta privada, solo se mostrarán cuando el usuario esté autenticado*/}
        <Route
          path="/*"
          element={
            <PrivateRoute>
              <DashboardRoute />
            </PrivateRoute>
          }
        />
        {/* <Route path="/*" element={<DashboardRoute />} /> */}
      </Routes>
    </BrowserRouter>
  );
};

export default AppRouter;
```

Rutas públicas
Crear otro componente para las rutas públicas:



```
import { useContext } from "react";
import { Navigate } from "react-router-dom";
import { AuthContext } from "../auth/authContext";

const PublicRoute = ({ children }) => {
  const { user } = useContext(AuthContext);

  return user.logged ? <Navigate to="/marvel" /> : children;
};

export default PublicRoute;
```

Renderizar en AppRouter:



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

const AppRouter = () => {
  return (
    <BrowserRouter>
      <Routes>
        {/* Login sin el Navbar */}
        {/* Ruta pública */}
        <Route
          path="/login"
          element={
            <PublicRoute>
              <LoginScreen />
            </PublicRoute>
          }
        />
        <Route path="/login" element={<LoginScreen />} />
        {/* Ruta privada, solo se mostrarán cuando el usuario esté autenticado*/}
        <Route
          path="/*"
          element={
            <PrivateRoute>
              <DashboardRoute />
            </PrivateRoute>
          }
        />

        {/* <Route path="/*" element={<DashboardRoute />} /> */}
      </Routes>
    </BrowserRouter>
  );
};

export default AppRouter;
```