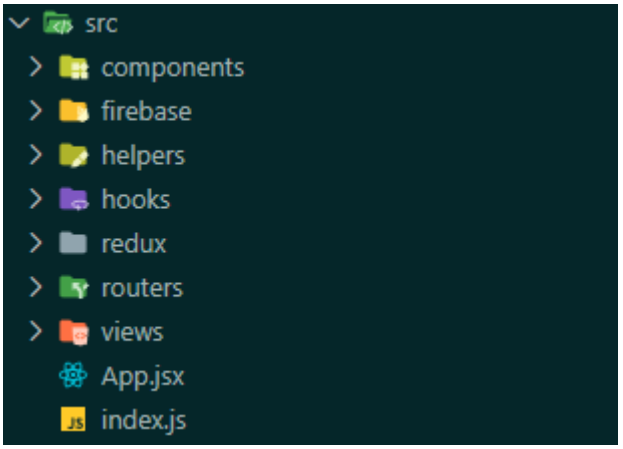
Documentació tècnica

GROUPEM

Albert Chao Vasco
Eric Clemente Casals
Arnau Orts Brichs

Institut Pedralbes

L'estructura de carpetes principal és la següent:



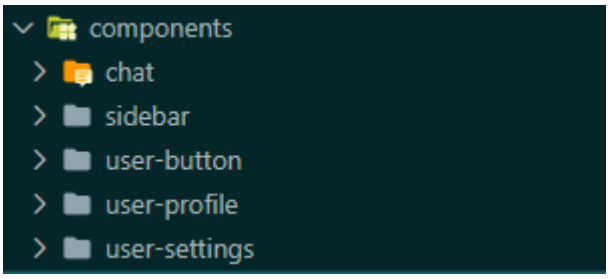
Els arxius "index" i "App" són els principals per tot projecte de React. Index és el primer arxiu que s'executa, aquest fa el render del component App, que pot contenir diferent contingut.
En el nostre cas cridem el component de "AppRouter", a més de tenir dos providers (Mantine i un store que s'utilitza per Redux).

```jsx
import React from "react";
import ReactDOM from "react-dom";
import App from "./App";

ReactDOM.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
  document.getElementById("root")
);
```

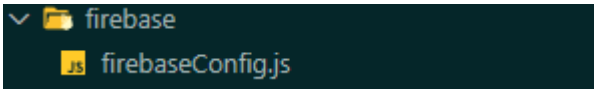
```jsx
function App() {
  return (
    <MantineProvider withNormalizeCSS>
      <Provider store={store}>
        <AppRouter />
      </Provider>
    </MantineProvider>
  );
}
```

Index a l'esquerra, App a la dreta.

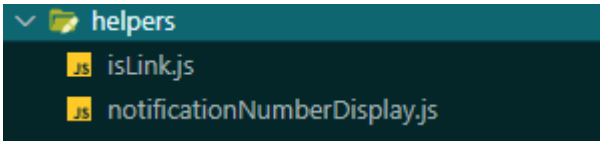
- “components” conté tots els components que són genèrics o que s’utilitzen no només a un lloc del projecte.



- “firebase” conté l’arxiu de configuració per poder connectar-se a Firebase.

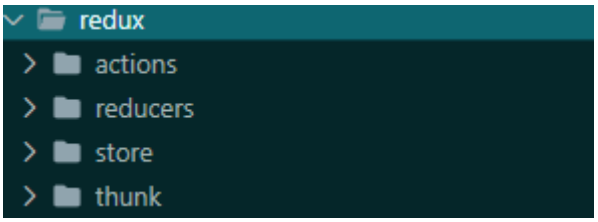


- “helpers” té alguns arxius que són funcions que s’utilitzen al projecte.

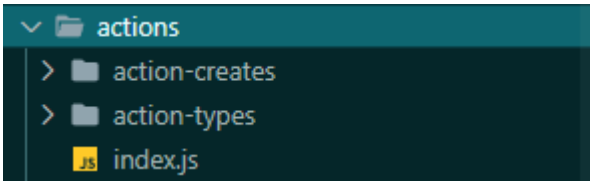


- “hooks” és una de les carpetes més importants quan es treballa amb React, tot i què no són obligatoris, conté components que han d’estar només un cop instanciats, en el nostre cas només utilitzem un.

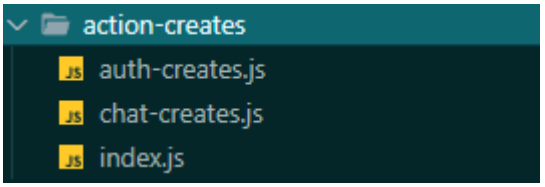
- Dins la carpeta “redux” trobem varies subcarpetes. Cada carpeta separa les quatre parts essencials.



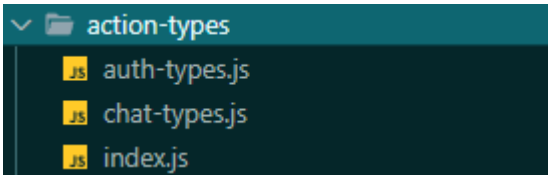
    - Dins la subcarpeta “actions“ podem trobar “action-creates” i “action-types”



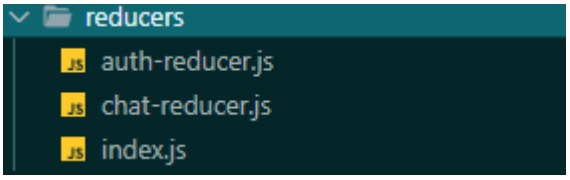
        - Els “action-creates” retornen l’acció corresponent amb el seu “type” i “payload”



        - Els “action-types” són tots els “types” utilitzats en els “action-creates”.

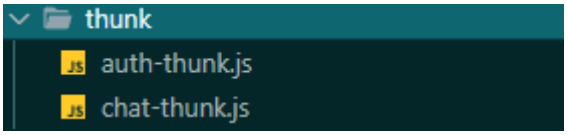


- Dins la subcarpeta “reducers” podem trobar tots els reducers que s’utilitzaran a la nostre store. La funció dels reduces és filtrar cada acció de l'usuari segons el “type” que rebi. Cada reducer rep com a paràmetre l’“state” i una acció.



- Dins de la subcarpeta “store” podem trobar l'arxiu store.js. Aquest arxiu s’encarrega de ajuntar tots els reducers creats i els combina com a un únic reducer, creant així una única font de la veritat. Gràcies a aquest store podem controlar en tot moment el “state” de la pàgina.

- Dins de la subcarpeta “thunk” podem trobar totes aquelles accions que abans de ser executades, necessiten fer trucades HTTP a altres serveis.

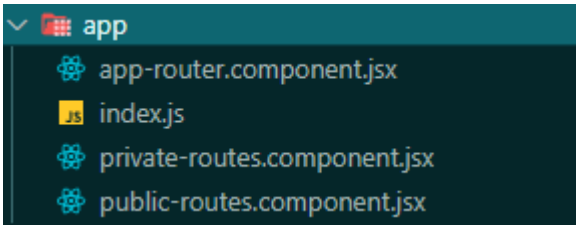


- “routers” conté tot el que té a veure amb les rutes que es manegen dins l’aplicació.

- Tenim un arxiu “paths” que té la informació de tots els paths de l’aplicació, per tenir-ho tot al mateix lloc.

```
const PATHS = {
  HOME: "/",
  ABOUT: "/about",
  CONTACT: "/contact",
  CHAT: "/chat",
  CHAT_ZONE: "*",
  LOGIN: "/login",
  REGISTER: "/register",
  //USER PROFILE ROUTES
  PROFILE_EDIT: "/profile/edit",
  PROFILE_USER: "/profile/:userId",
  SETTINGS: "/settings",
  //GROUPS ROUTES
  GROUP_FINDER: "/group-finder",
  GROUP_CREATE: "/group/create",
  GROUP_DETAILS: "/group/:groupId",
  GROUP_EDIT: "/group/:groupId/edit",
  //ERROR
  NOT_FOUND: "/error-404",
};
```

- Dins la subcarpeta “app” hi ha els components per les rutes públiques, privades i el component app-router, que fa de controlador de rutes.



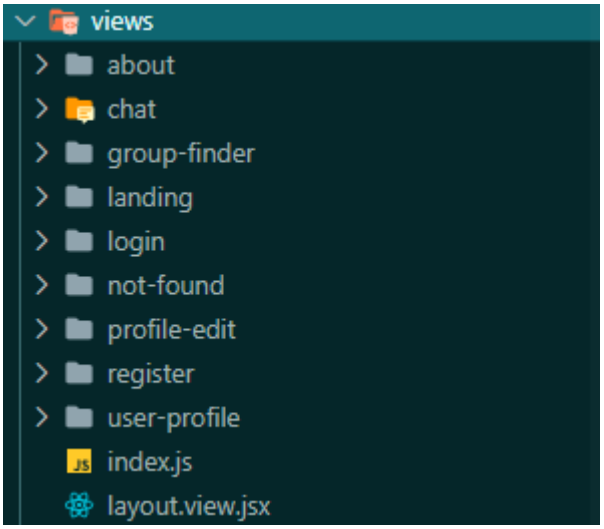
Els components de “private-routes” i “public-routes” només fan el control de si el path on es vol anar és privat o públic, i canvien la ruta si l’usuari no té els permisos.
Un exemple és, si l’usuari ja està autenticat no podrà entrar a les rutes públiques (login i registre) mentre que un usuari no autenticat podrà entrar a les rutes públiques però no a les privades.

Un exemple de l'arxiu "app-router", controlador de rutes:

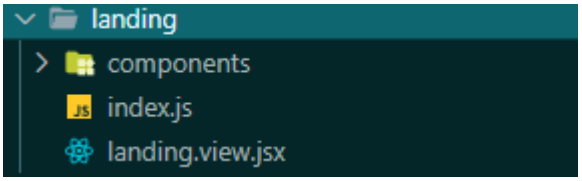
```jsx
<Routes>
  {/* Generic routes */}
  <Route path={PATHS.HOME} element={<Landing />} />
  <Route path={PATHS.ABOUT} element={<About />} />

  {/* Public routes */}
  <Route
    path={PATHS.LOGIN}
    element={
      <PublicRoutes isLogged={isAuthenticated}>{<Login />}</PublicRoutes>
    }
  />
  <Route
    path={PATHS.REGISTER}
    element={
      <PublicRoutes isLogged={isAuthenticated}>
        {<Register />}
      </PublicRoutes>
    }
  />

  {/* Private routes */}
  <Route
    path={PATHS.CHAT}
    element={
      <PrivateRoutes isLogged={isAuthenticated} navbarType={"chat"}>
        <Chat />
      </PrivateRoutes>
    }
  />
```

Les rutes genèriques són les que qualsevol usuari registrat o no pot entrar.

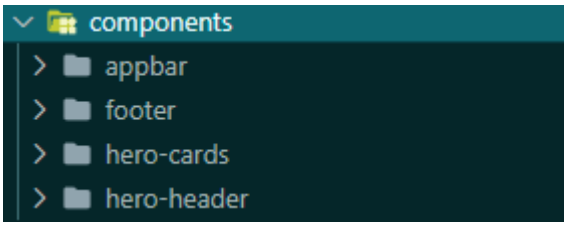
- La carpeta "views" és la més extensa de tot el projecte, ja que conté totes les vistes que crida el router.



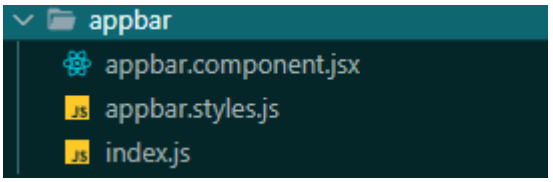
- Per cadascuna de les vistes tenim una carpeta, només explicarem una ja que totes tenen la mateixa estructura.
  La vista de “landing”, per exemple, té el component que és el que s’executa i una carpeta “components”, que conté tots els components interns que només s’utilitzen per aquesta vista i tenir una bona organització.



  - Cada subcomponent té la seva carpeta, totes segueixen la mateixa estructura també.

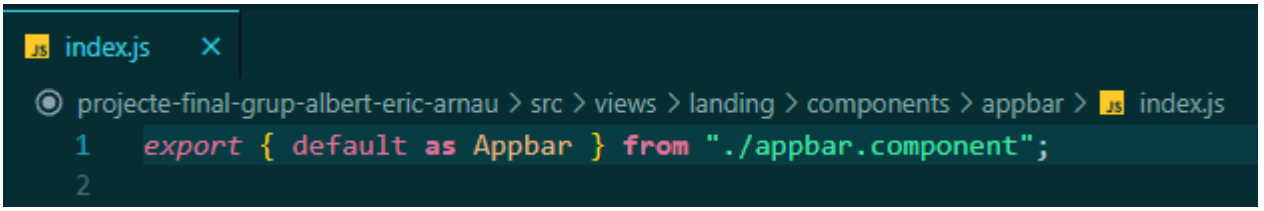


  - Els subcomponents tenen, a més del propi component, un arxiu d’estils per així tenir separat la part de visual i la part de funcionalitat.



Per acabar, com s’ha pogut veure, quasi bé totes les carpetes tenen un arxiu “index”. Aquest permet fer d’una forma més fàcil i entenedora les importacions i exportacions, a més de millorar tota l’organització del projecte.

Un exemple d’un arxiu index és del recent subcomponent “appbar”.



Únicament conté l’exportació de l’arxiu “appbar”.

Per tant, al component landing s’ha d’importar de la següent manera, fent referència a l’arxiu index.