Hola David muy buenas tardes, espero que hayas empezado con buen el 2020 😊.

Ya he rehecho la práctica de React y he ido siguiendo punto a punto las cosas que me faltaban según el feedback que me diste. Para facilitarte la corrección, te copio los puntos del feedback y te voy poniendo mis comentarios en cada punto. Te informo de que me ha venido muy bien la solución que nos pasaste y he cogido muchas de tus ideas que he ido amoldando a mi práctica (como lo de monitorizar en el store la última acción, o el proteger las rutas con el componente ProtectedRoutes, entre otras):

* No se ha configurado redux thunk en el store. Las acciones de carga de datos se manejan fuera de redux. Funciona, pero no es lo que se pedía. La carga de anuncios en la home funciona por el truco que has hecho de fetchAdverts()(dispatch, params), pero no es la manera. De hecho la acción fetchAdverts no está configurada correctamente para ser lanzada por el middleware de thunk.

**Ya he configurado redux think en el store. Además, he metido todas las ejecuciones de los servicios de la API dentro de las acciones asíncronas para que se manejen desde Redux.**

* No se lee el user del localStorage al configurar el store. Una vez leído del localStorage y cargado en redux el usuario, no debería usarse para nada más el localStorage. Este debería usarse solo para persistir lo que haya en redux, pero se está usando actualmente como fuente de verdad en algunos componentes. Por ejemplo en AdvertDetail se lee el user de redux y localStorage. Además, el AdvertDetail se encarga de loguear al usuario a Redux si no lo está. Es muy lioso. Al final esa lógica está muy repetida en varios componentes.

**He quitado la llamada al localStorage de todos los componentes y la hace solo una vez al arrancar la aplicación y al configurar el store.**

**También he usado HOCs como lo tenías tú (withAdverts, withSession, etc) para conectar los componentes, y así no repetir código en cada index.js.**

* El componente Home hace muchas cosas, llamadas al local storage en el render, llamadas a history.push…

**Lo he limpiado y ahora solo hace lo que tiene que hacer.**

* Los componentes están conectados a redux en el mismo fichero donde se declaran, se hace difícil de seguir.

**He separado la lógica de los componentes metiendo las conexiones a los HOCs y Redux en el index.js dejando el componente más limpio para que haga solo lo suyo.**

* Los redirects tras acciones (login, logout), mejor hacerlos en las acciones en lugar de en los componentes.

**Ahora tanto el login como el logout hacen el redireccionamiento ya que les paso el “history” en cada acción (como proponías en tu solución).**

* El guardado en el local storage mejor centralizarlo en la acción de login o con un subscribe al store.

**Lo hago cada vez que se crea el store, y mediante un subscribe lo vuelvo a guardar cada vez que sufre un cambio. He cogido tú idea de estar siempre monitorizando la última acción ejecutada y si es del tipo login actualizo el localStorage, y si es del tipo logout lo vacío.**

* Hay llamadas a servicios en componentes

**Como comentaba en otro punto más arriba, he centralizado todas las llamadas a los servicios de la API en las acciones, y las he quitado de los componentes.**

* Los tags se podrían meter en redux en redux y se cargarían del api una sola vez.

**Los tags ahora se cargan una sola vez cuando se inicia la aplicación y los persisto en el state.**

* Los nombres de las partes del state no son muy afortunados, procura evitar que se llamen userReducer, advertReducer. Mejor user, advert…

**He eliminado el “reducer” y los he dejado como comentas.**

* Pasan los test aunque hay un warning en el test de App porque está intentando hacer un render complete de la app, sin haber configurado un store.

**He quitado el warning y he modificado todos los test para que pasen. He testeado las acciones de carga de anuncios y de logado, los reducers de lo mismo y los componentes Home y App.**

* El reactor del formulario no está hecho

**Esta parte es la que más me ha costado entender. He tenido que tirar mucho de tu solución para poder terminar esta parte.**

**Lo he usado solo en el Registro ya que es el único que usaba un form como tal. En el resto uso Forms de Material UI. Ya he aprovechado y he refractorizado el componente entero y lo he pasado a function con hooks.**