


Open

Opened 5 days ago by  **Rubén Montero**

Desloguearse

Resumen

- Enviaremos un HTTP DELETE para *desloguearnos* al clicar en *desloguearte* en `Container.js`
- También borraremos los datos de `sessionStorage` y haremos `setLoggedIn(false)`

Descripción

Para finalizar nuestro maravilloso proyecto, permitamos al usuario *desloguearse*.

La tarea

Añade un método `sendLogoutRequest` a `Container.js` que envíe la siguiente petición DELETE:

```
const sendLogoutRequest = () => {
  const id = sessionStorage.getItem('SESSION_ID');
  const token = sessionStorage.getItem('SESSION_TOKEN');
  if ((id !== null) && (token !== null)) {
    axios.delete('http://raspi:8082/api/v2/sessions/'+id, { headers: {'Session-Token': token } })
  }
}
```

Luego, **invócala** desde `doLogout` . También, **completa** el proceso de *logout* **así**:

```
const doLogout = () => {
  sendLogoutRequest()
  // Logout lado cliente
  sessionStorage.removeItem('SESSION_ID');
  sessionStorage.removeItem('SESSION_TOKEN');
  setLoggedIn(false);
}
```

Como curiosidad, verás que hemos decidido no esperar a la respuesta del HTTP DELETE. ¿Para qué hacerlo? Si esperamos a la confirmación del servidor de que nos hemos *deslogueado* para hacer `sessionStorage.removeItem` y `setLoggedIn(false)` *podríamos* estar dejando al usuario bloqueado sin poder *desloguearse* en caso de que la petición fallase incontroladamente.

Haciéndolo como lo hemos hecho (sin esperar al servidor) garantizamos que nos *deslogueamos* desde el cliente. Como desventaja, si el HTTP DELETE falla, el servidor no sabrá que esa sesión ya se ha perdido.

¿Comprendes las posibilidades? ¿Cuál te parece mejor?

Y... ¡proyecto terminado!


Has completado `react-stackoverrun` .

¡Felicidades!

Comprueba su funcionamiento en <http://localhost:3000>. ¿Qué te parece el resultado?

Por último

Sube tus cambios al repositorio en un nuevo *commit*.



[Rubén Montero @ruben.montero](#) changed milestone to [%Sprint 5](#) 5 days ago