Open Opened 5 days ago by **Rubén Montero**

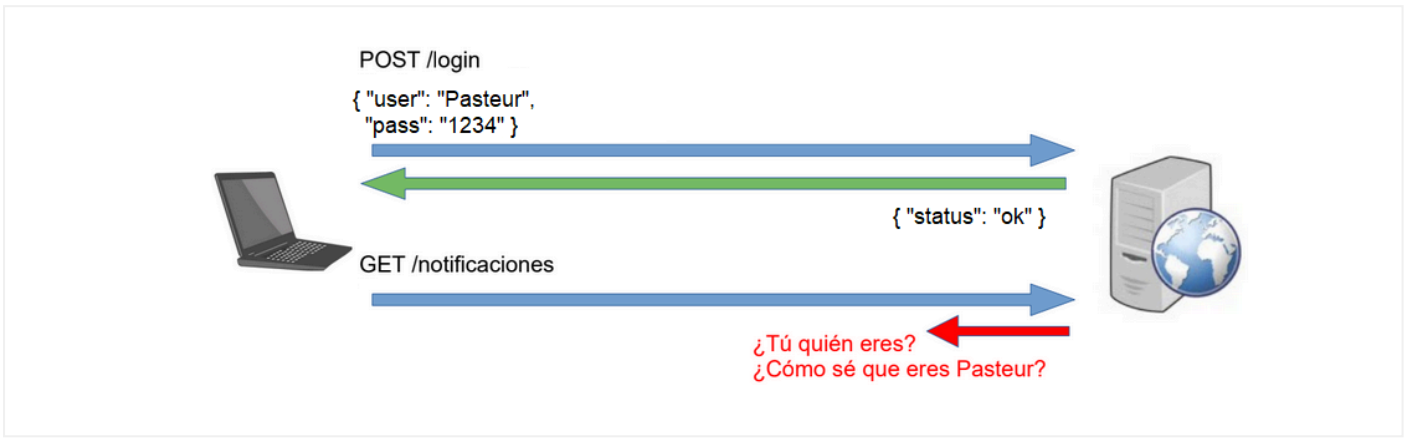
Un formulario de login bonito... y funcional

Resumen

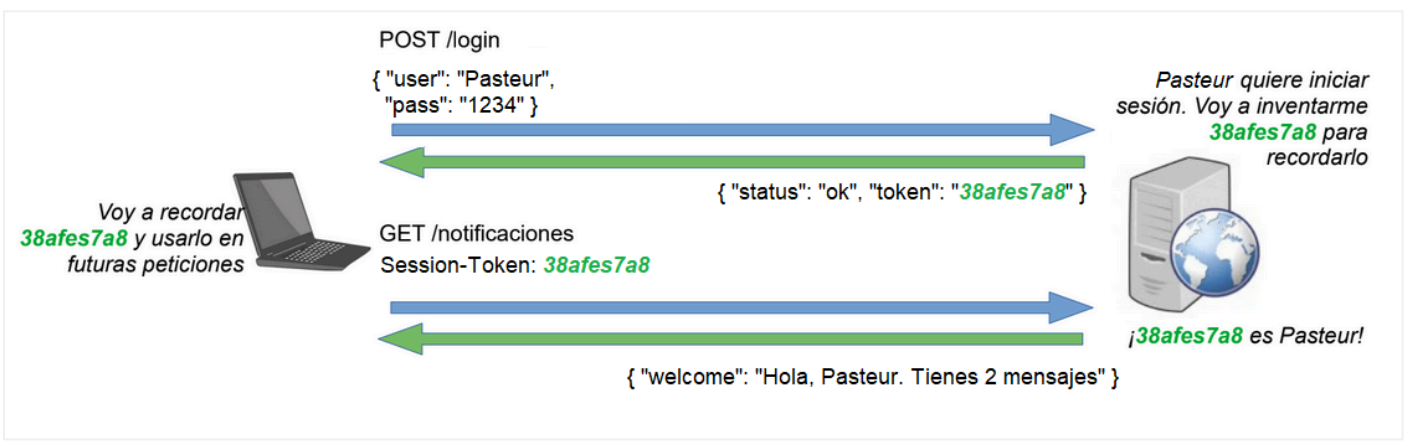
- Hablaremos de un flujo de autenticación a través de un *token* de sesión
- Enviaremos la petición POST necesaria cuando se clica en *Login*
- Si el servidor responde con éxito, guardaremos el *token* con `sessionStorage` y redirigiremos al usuario a la página principal

Descripción

Las APIs REST (REpresentational State Transfer) no almacenan estado del cliente. Toda la información relevante viaja en la petición. Es decir, no podemos esperar que el servidor REST *nos recuerde*



Para abordar el concepto de *sesión*, en el ecosistema HTTP, se suele emplear la siguiente aproximación:



Aunque este flujo puede suceder de manera más *automática* (entre el navegador y el servidor) si se configura autenticación mediante *cookies*, nosotros vamos a implementarlo más *manualmente* y a gestionar, desde nuestro código React:

1. La recepción del *token* (tras un *login*)
2. Guardarlo
3. Enviarlo en peticiones que requieren autenticación

En esta tarea, nos centraremos en (1) y (2).

La tarea

Añade `onSubmit={onSubmit}` a la etiqueta `<form>` en `UserLogin.js` .

Después, **añade** dicho método como se indica a continuación:

```
const onSubmit = (e) => {
  e.preventDefault()
  if ((username.length === 0) || (password.length === 0)) {
    return;
  }
  axios.post('http://raspi:8082/api/v2/sessions', { username: username, password: password }).then(response => {
    console.log(response.data.session_token);
    console.log(response.data.session_id);
  });
```

Como ves, recibimos el *token* de sesión (y también un *id*; que será relevante para desloguearnos).

Ahora, hay que almacenarlos en el navegador.

Web Storage API

Los navegadores tienen una funcionalidad Web Storage. Nos otorga dos objetos:

- `localStorage` : Almacenamos datos en la caché del navegador. Estos datos se mantienen ahí hasta que se borra la caché.
- `sessionStorage` : Almacenamos datos en la *sesión* de la pestaña del navegador. Dichos datos no son compartidos con otras pestañas del navegador. Cuando cierras la pestaña actual, se borran automáticamente.

Para nuestro caso de uso, `sessionStorage` es más adecuado[1].

Añade estas líneas al `onSubmit` :

```
    axios.post('http://raspi:8082/api/v2/sessions', { username: username, password: password }).then(response => {
      console.log(response.data.session_token);
      console.log(response.data.session_id);
+     sessionStorage.setItem('SESSION_TOKEN', response.data.session_token);
+     sessionStorage.setItem('SESSION_ID', response.data.session_id);
    });
```

Y, tras **importar** `useNavigate` (`const navigate = useNavigate();`) como ya has hecho anteriormente, **redirige** al usuario a la página principal:

```
    axios.post('http://raspi:8082/api/v2/sessions', { username: username, password: password }).then(response => {
      console.log(response.data.session_token);
      console.log(response.data.session_id);
      sessionStorage.setItem('SESSION_TOKEN', response.data.session_token);
      sessionStorage.setItem('SESSION_ID', response.data.session_id);
+     navigate('/');
    });
```

¡Enhorabuena! Has implementado una página de *login*

🏆 Por último

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

---

1. `sessionStorage` no es óptimo para guardar datos de sesión, pues es vulnerable a ataques XSS. Lo ideal es emplear *Cookies* de sesión, donde, mediante cabeceras HTTP, la sesión es gestionada por el navegador automáticamente. Inicialmente, por razones didácticas, trabajaremos con los datos de sesión *a mano*. ↩

Rubén Montero @ruben.montero changed milestone to %Sprint 5 5 days ago