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

Un formulario de registro bonito

Resumen

- Crearemos `UserRegister.js`
- Añadiremos un formulario de registro con 3 campos: *Nombre de usuario*, *Contraseña* y *Confirmar contraseña*, enlazados a sus respectivos estados
- Completaremos el formulario con algo de CSS. Todavía *no* será funcional

Descripción

Vamos a trabajar el registro y autenticación de usuarios.

El *registro de un usuario* consiste en añadir una nueva fila a la base de datos. Nuestro API REST tiene un *endpoint* dedicado a ello:

- POST http://raspi:8082/api/v2/users

Para ir poco a poco, en esta tarea crearemos un formulario de registro *todavía no funcional*.

La tarea

Crea una nueva pantalla `src/screens/user/register/UserRegister.js` .

En `UserRegister.js` , **añade** los siguientes estados:

```
const [username, setUsername] = useState('');
const [password, setPassword] = useState('');
const [passwordConfirm, setPasswordConfirm] = useState('');
```

En la parte visual del componente (`return`), **añade** un `<div>` principal, y dentro:

- Una etiqueta `<h1 data-cy=pageHeader>` que muestre `Crea tu cuenta`
- Un `<form data-cy=userRegisterForm>` que contenga:
  - Un primer `<input>` con:
    - `data-cy=inputUsername`
    - `type=text`
    - `onChange={onChangeUsername}` *(La escribiremos a continuación)*
    - `placeholder='new.user'`
    - `value={username}`
  - Después, una etiqueta `<br />`
  - Un segundo `<input>` :
    - `data-cy=inputPassword`
    - `type=password`
    - `onChange={onChangePassword}` *(La escribiremos a continuación)*
    - `placeholder='Contraseña'`
    - `value={password}`
  - Después, una etiqueta `<br />`
  - Un tercer `<input>` :
    - `data-cy=inputPasswordConfirm`
    - `type=password`
    - `onChange={onChangePasswordConfirm}` *(La escribiremos a continuación)*
    - `placeholder='Confirmar contraseña'`
    - `value={passwordConfirm}`
  - Después, una etiqueta `<br />`
  - Un cuarto `<input>` :
    - `data-cy=inputSubmit`
    - `type=submit`
    - `value='Empezar'`

A continuación, **añade** las funciones necesarias:

```
const onChangeUsername = (e) => {
  setUsername(e.target.value);
}

const onChangePassword = (e) => {
  setPassword(e.target.value);
}

const onChangePasswordConfirm = (e) => {
  setPasswordConfirm(e.target.value);
}
```

¡Veamos nuestro formulario!

Añade en `App.js` :

```
      <Route path='/register' element={<UserRegister/>}/>
```

Y ahora ya estará visible la página http://localhost:3000/register

¿Se ve bien?

Un poco de CSS

Crea un nuevo fichero `src/screens/user/register/UserRegister.css` con el siguiente CSS[1]:

```
.registerHeader, .registerForm {
  padding-top: 42px;
  text-align: center;
}

.registerFormInput {
  height: 48px;
  background-color: rgba(57, 57, 57, 0.08);
  border: none;
  border-radius: 11px;
  margin: 5px;
  padding: 0px 20px;
}

.registerFormInput:hover {
  background-color: rgba(57, 57, 57, 0.20);
}
```

Para terminar, en `UserRegister.js` **añade** el `import` :

```
import ./UserRegister.css
```

Y **añade** también los `className` necesarios:

- `<h1>` → `className="registerHeader"`
- `<form>` → `className="registerForm"`
- `<input>` *(los 4)* → `className="registerFormInput`

¿Qué tal se ve http://localhost:3000/register ahora?

🏆 Por último

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

---

1. Si te preguntas: *¿Como se puede hacer un buen diseño usando CSS?* la respuesta es: Practicando. Los navegadores ofrecen la útil herramienta *Click derecho > Inspeccionar* que te permite entender *qué* CSS exactamente está afectando a un elemento. Busca elementos similares en págines de Internet que te resulten atractivos e inspecciónalos. Basta con copiar y usar su CSS en tu página, intentando mantenerlo sencillo (principio KISS). ⮌

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