

El objeto de esta práctica es afianzar, reforzar y consolidar los conocimientos teóricos presentados en la lección.

Al finalizarla, el estudiante:

- Habrá creado componentes formularios.
- Habrá creado componentes de entrada de datos `<input>`, `<textarea>` y `<select>`.

Objetivos

El objetivo de la práctica es mostrar cómo trabajar con formularios en **React**. Para ello, crearemos formularios y elementos `<input>`, `<textarea>`, `<select>` y `<option>`.

Preparación del entorno

Para comenzar, crearemos un proyecto de aplicación **React** mediante el generador de **Justo**:

1. Abrir una consola.
2. Crear el directorio de la práctica e ir a él.
3. Mostrar la ayuda del generador:

```
> justo -g react help
```
4. Invocar el generador de **Justo**:

```
> justo -g react
```

No es necesario **React Router**.
5. Instalar las dependencias del proyecto:

```
> npm install
```
6. Mostrar el catálogo de tareas automatizadas del proyecto:

```
> justo -c
```
7. Invocar la tarea **build** del catálogo del proyecto para construir la aplicación:

```
> justo build
```
8. Abrir el archivo `dist/index.html` con el navegador.
Debe aparecer el mensaje Hello World!

Generación del componente formulario

Primero, vamos a crear el componente formulario que tiene como objeto recopilar las credenciales de autenticación de un usuario:

1. Ir a la consola.
2. Crear un componente formulario mediante el generador:

```
> justo -g react form
```

A la hora de responder:

 - Indicar `<none>` como subcarpeta de `app/components` donde añadirlo.
 - Indicar `LoginForm` como nombre del componente.
 - Indicar **Y** cuando pregunte si cancelar la acción predeterminada de `<form>` en el evento `Submit`.
 - Indica **Y** cuando pregunte si definir el atributo `onChange` a nivel de formulario.

3. Editar el archivo `app/components/LoginForm.jsx`. Y echarle un vistazo.

4. Ir al constructor e indicar el estado inicial como sigue:

```
constructor(props) {  
  super(props);  
  
  this.state = {  
    username: "",  
    password: ""  
  };  
}
```

5. Modificar el controlador de evento `handleSubmit()` para que se muestre la información del estado del formulario cuando el usuario haga clic en el botón de envío:

```
handleSubmit(evt) {  
  evt.preventDefault();  
  alert(`Usuario: ${this.state.username}\nContraseña: ${this.state.password}`);  
}
```

Se mantiene la cancelación de la acción predeterminada para que el formulario no sea enviado al servidor.

6. Guardar cambios.

7. Editar el archivo `app/views/App.jsx`.

8. Importar el componente formulario:

```
import LoginForm from "../components/LoginForm";
```

9. Reproducir el formulario en el método `render()`:

```
render() {  
  return (  
    <LoginForm />  
  );  
}
```

10. Guardar cambios.

Generación de componentes de entrada de datos

Ahora, vamos a generar los componentes `<input>` con los que solicitar el nombre de usuario y la contraseña, así como el botón con el que solicitar el envío de los datos al servidor.

1. Ir a la consola.

2. Generar el código de un componente `<input>` mediante el generador:

```
> justo -g react snippet input
```

Responder teniendo en cuenta lo siguiente:

- El valor del atributo `type` debe ser `text`.
- El nombre del elemento, atributo `name`, debe ser `username`.
- El tipo de componente debe ser `Controlled`.
- El subtipo de componente controlado debe ser `Smart`.
- El valor de `value` debe ser `{this.state.username}`.
- No hay que añadir el atributo `onChange`, pues lo gestionaremos a nivel de formulario.
- Añadir el atributo `autoFocus`.
- Añadir el atributo `required`.
- Indicar el texto `Your username` en el atributo `placeholder`.

El código del elemento aparecerá en la consola. Algo como lo siguiente, pero puede cambiar según la versión del generador:

```
<input type="text" name="username" value={this.state.username} placeholder="Your  
username" required autoFocus />
```

Si además tiene el comando correspondiente instalado en el sistema, es posible que también se encuentre en el portapapeles. En **Windows**, el comando que usa el generador para copiar en el portapapeles es **clip**; y en **Linux**, **xclip**.

3. Editar el archivo **app/components/LoginForm.jsx**.
4. Hacer **Ctrl+C** para pegar el código del componente del portapapeles en el formulario:

```
render() {  
  return (  
    <form onSubmit={this.handleSubmit}>  
      <input type="text" name="username" ... />  
    </form>  
  );  
}
```

Si no lo tiene en el portapapeles, vuelva a la consola, cópielo y péguelo en el **<form>**.

5. Volver a la consola.
6. Generar el código de otro componente **<input>** mediante el generador:

```
> justo -g react snippet input  
Ahora:
```

- Tipo de **<input>**: **password**. Desplácese con las flechas arriba y abajo hasta encontrarlo.
- Nombre de elemento: **password**.
- Tipo de componente: **Controlled**.
- Tipo de componente controlado: **Smart**.
- Valor: **{this.state.password}**.
- **onChange**: **N**, pues hemos decidido usar el controlador definido a nivel de formulario.
- **autoFocus**: **N**.
- **required**: **Y**.
- **placeholder**: **Your password**.

7. Ir al archivo **app/components/LoginForm.jsx**.
8. Pegarlo debajo del componente **username**.
9. Ir a la consola.
10. Invocar el generador para crear un **<input>** nuevo.

Esta vez:

- Tipo de **<input>**: **submit**.
- Texto a mostrar: **Sign in**.

11. Ir al archivo **app/components/LoginForm.jsx**.
12. Pegarlo debajo del componente **password**.
13. Echar un vistazo al método **handleChange()** con el que se almacenará los cambios en el estado del formulario. Tómese su tiempo.
14. Guardar cambios.
15. Construir y abrir la aplicación en el navegador:

```
> justo build chrome
```

Debe aparecer algo como:



The screenshot shows a web form with two input fields. The first field is labeled "Your username" and the second field is labeled "Your password". To the right of the second field is a button labeled "Sign in". The "Your username" field is currently selected with a blue border.

16. Escribir un nombre de usuario y una contraseña.
17. Hacer clic en el botón **Sign in**.

Debe aparecer algo como:

This page says:

Usuario: elvis

Contraseña: costello

OK

Este mensaje lo genera el controlador del evento `Submit`, recordemos, `handleSubmit()`. Échele un vistazo y observe que los datos los extrae del estado. El cual se ha fijado mediante el controlador del evento `Change`, `handleChange()`.

Generación de otros tipos de componentes

Una vez ya sabemos cómo crear el componente formulario y los elementos `<input>`, vamos a practicar con otro tipos de elementos de entrada. Para ello, vamos a crear un formulario de contacto con el que el usuario puede dejar un mensaje.

1. Ir a la consola.
2. Crear un nuevo formulario `ContactForm` con el generador.

Hacerlo de manera similar a como creamos `LoginForm`.

3. Editar el archivo `app/components/ContactForm.jsx`.
4. Definir el estado inicial a:

```
constructor(props) {  
  super(props);  
  
  this.state = {  
    email: "",  
    lang: "",  
    veryImportant: false,  
    message: ""  
  };  
}
```

5. Modificar el método controlador del evento `Submit` para que muestre la información:

```
handleSubmit(evt) {  
  evt.preventDefault();  
  alert(`Correo: ${this.state.email}\nIdioma de contacto: ${this.state.lang}\nMensaje importante: ${this.state.veryImportant}\nMensaje: ${this.state.message}`);  
}
```

6. Ir a la consola.
7. Crear un componente de entrada de datos que recolecte una dirección de correo electrónico:
 - Tipo de `<input>`: `email`.
 - Nombre de elemento: `email`.
 - Tipo de componente: `Controlled`.
 - Tipo de componente controlado: `Smart`.
 - Valor: `{this.state.email}`.
 - `onChange`: `N`.
 - `autoFocus`: `Y`.
 - `required`: `Y`.
 - `placeholder`: `Your email`.
8. Ir al formulario `ContactForm`.
9. Pegarlo en el formulario.

10. Ir a la consola.
11. Generar un elemento `<select>` para que el usuario indique el idioma en el que desea ser contactado:
> justo -g react snippet select
Responda como sigue:
 - Nombre de elemento: lang.
 - Tipo de componente: **Controlled**.
 - Tipo de componente controlado: **Smart**.
 - **onChange**: N.
 - **selected**: {this.state.lang}.
 - **required**: Y.
 - **multiple**: N.
12. Ir al formulario ContactForm y pegarlo debajo del componente lang.
13. Ir a la consola.
14. Generar las opciones del `<select>`: una vacía, otra para español, otra para inglés y otra para italiano. Para ello, usar el generador:
> justo -g react snippet option
Después de crear cada una de ellas, ir al formulario y pegarlo dentro del `<select>`. Debe quedar algo como sigue:

```
<select name="lang" selected={this.state.lang} required>
  <option value=""></option>
  <option value="es">Spanish</option>
  <option value="en">English</option>
  <option value="it">Italian</option>
</select>
```
15. Ir a la consola.
16. Generar un cuadro de confirmación para que el usuario pueda indicar si se trata de un mensaje importante. Haga uso del generador:
> justo -g react snippet input
Conteste teniendo en cuenta lo siguiente:
 - Tipo de `<input>`: **checkbox**.
 - Nombre de elemento: **veryImportant**.
 - Texto a mostrar: **Very important**.
 - Delimitarlo mediante un `<label>`: **Y**.
 - Tipo de componente: **Controlled**.
 - Tipo de componente controlado: **Smart**.
 - **checked**: {this.state.veryImportant}.
 - **onChange**: N.
17. Ir al formulario y pegarlo debajo del componente lang.
18. Volver a la consola.
19. Crear un elemento `<textarea>` mediante el cual recoger el mensaje del usuario:
> justo -g react snippet textarea
Recolección de datos del generador:
 - Nombre del elemento: **message**.
 - Tipo de componente: **Controlled**.

- Tipo de componente controlado: **Smart**.
- Valor: `{this.state.message}`.
- **onChange**: **N**.
- **autoFocus**: **N**.
- **required**: **Y**.
- **placeholder**: Your message.

20. Ir al formulario y pegarlo debajo del elemento `veryImportant`.

21. Crear el botón de envío de datos:

```
> justo -g react snippet input
```

Respuestas:

- Tipo de `<input>`: **submit**.
- Texto a mostrar: Send.

22. Copiarlo en el formulario debajo del elemento `message`.

El formulario debería quedar más o menos como sigue:

```
<form onSubmit={this.handleSubmit}
  onChange={this.handleChange}>
  <input type="email"
    name="email"
    value={this.state.email}
    placeholder="Your email"
    required
    autoFocus />
  <select name="lang" selected={this.state.lang} required>
    <option value=""></option>
    <option value="es">Spanish</option>
    <option value="en">English</option>
    <option value="it">Italian</option>
  </select>
  <label>
    <input type="checkbox"
      name="veryImportant"
      checked={this.state.veryImportant} />
    Very important
  </label>
  <textarea name="message"
    value={this.state.message}
    placeholder="Your message"
    required />
  <input type="submit" value="Send" />
</form>
```

23. Guardar cambios.

24. Editar el archivo `app/views/App.jsx`.

25. Importar el formulario:

```
import ContactForm from "../components/ContactForm";
```

26. Modificar el método `render()`:

```
render() {
  return (
    <ContactForm />
  );
}
```

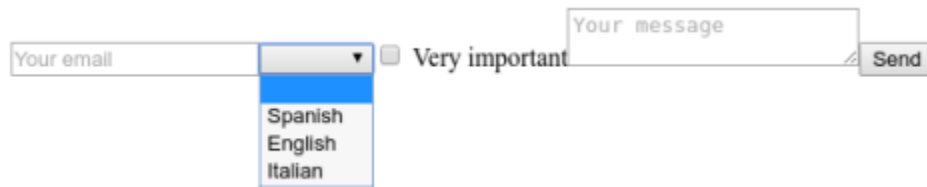
27. Guardar cambios.

28. Ir a la consola.

29. Reconstruir y abrir la aplicación:

```
> justo build chrome
```

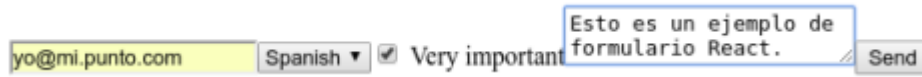
Debe aparecer algo como lo siguiente:



Formulario de correo electrónico:

- Campo "Your email" con un menú desplegable que muestra "Spanish", "English" y "Italian".
- Campo "Very important" con un icono de bandera.
- Campo "Your message" con el texto "Esto es un ejemplo de formulario React."
- Botón "Send".

30. Rellenar el formulario:

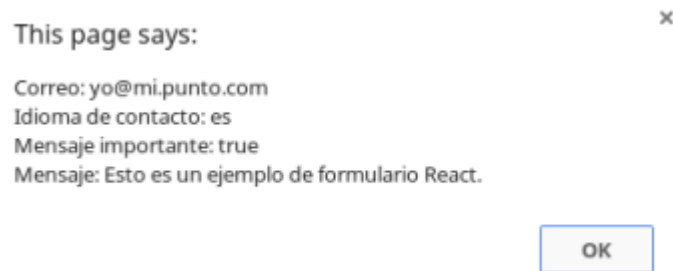


Formulario de correo electrónico relleno:

- Campo "Your email" con el valor "yo@mi.punto.com".
- Campo "Spanish" con un icono de bandera.
- Campo "Very important" con un icono de bandera.
- Campo "Your message" con el texto "Esto es un ejemplo de formulario React."
- Botón "Send".

31. Hacer clic en el botón Send.

Debe aparecer la información introducida:



Esta página dice:

- Correo: yo@mi.punto.com
- Idioma de contacto: es
- Mensaje importante: true
- Mensaje: Esto es un ejemplo de formulario React.

Botón "OK".