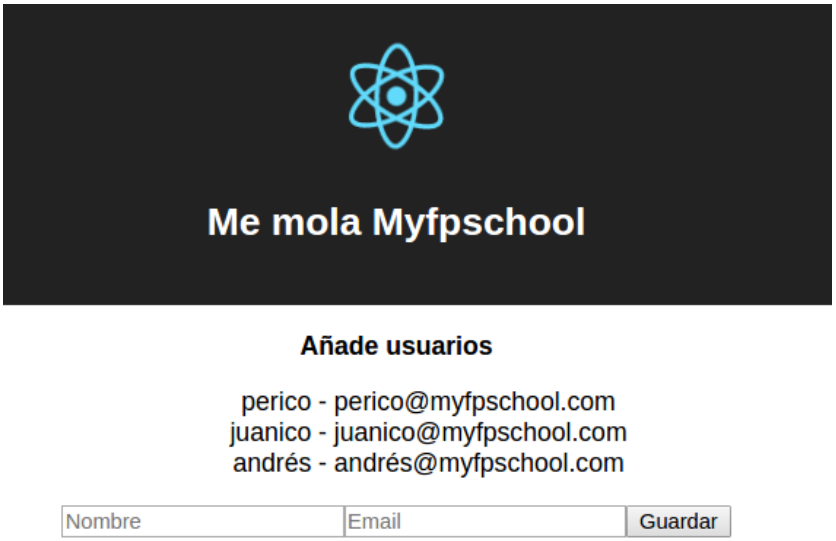
EJERCICIO 4

Se pide realizar una aplicación que registre usuarios y emails con un formato como el siguiente:



PASO 0 CREAR LA APLICACIÓN CON CREATE-REACT-APP

El primer paso que hay que hacer es crear la aplicación con el programa create-react-app tal y como se ha explicado anteriormente.

Para esta aplicación, en todo momento trabajaremos con el fichero App.js que es donde reside el componente principal de la misma. Cuando creamos una aplicación con el

programa anterior, el componente App es un fuction component el cual deberémos transformar a un Class component porque le vamos a añadir estado y los function components solamente pueden almacenar estado utilizando hooks (más concretamente el hook useState).

Por lo tanto, una vez creada la aplicación eliminaremos lo innecesario y cambiaremos el function component App por un class component de la siguiente forma:

```
import React, { Component } from 'react';
import logo from './logo.svg';
import './App.css';
class App extends React.Component {
 render() {
   return (
     <div className="App">
       <div className="App-header">
         <img src={logo} className="App-logo" alt="logo" />
         <h2>Me mola Myfpschool</h2>
       </div>
       <div>
         <p><strong>Añade usuarios</strong></p>
       </div>
     </div>
   );
 }
}

export default App;
```

Consejo: Prueba tras estos cambios que funciona tu componente y que se renderiza correctamente.

PASO 1. CREANDO EL CONSTRUCTOR DEL COMPONENTE APP

Fichero App.js: Es el momento de que nuestro componente tenga estado y por lo tanto vamos a crear un constructor con la variable state dentro de él. Deberás añadir el constructor al componente App utilizando el código siguiente:

```
 constructor() {
```

```
    super();
    this.state = {
      users: [
        {id: 1, name: "perico", email: "perico@myfpschool.com"},
        {id: 2, name: "juanico", email: "juanico@myfpschool.com"},
        {id: 3, name: "andrés", email: "andrés@myfpschool.com"}
      ]
    };
  }
```

Como puedes ver en el código, hemos creado varios usuarios por defecto que nos vendrán muy bien para testear nuestra aplicación. Al final del todo si lo crees oportuno los puedes eliminar.

PASO 2. CREANDO LOS COMPONENTES USER Y USERLIST

Vamos a crear dos componentes llamados **User** y **UserList**, los cuales los vamos a ubicar en dos archivos diferentes: User.js y UserList.js.

El componente User mostrará un usuario con el formato que se ha propuesto anteriormente.

FICHERO User.js

```
import React, { Component } from 'react';
class User extends Component {
 render() {
    return(
      <li>
      {this.props.name} - {this.props.email}
      </li>
    );
 }
}
export default User;
```

Como el componente User muestra solo un usuario, necesitaremos otro componente superior que muestre toda la lista de usuarios disponible. Para ello crearemos el componente UserList.

FICHERO UserList.js

```
import React, { Component } from 'react';
import User from './User';

class UserList extends Component {
 render() {
   return(
     <ul>
     {this.props.users.map(u => {
         return (
             <User
               key={u.id}
               name={u.name}
               email={u.email}
             />
           );
         })
       }
     </ul>
   );
 }
}
export default UserList;
```

Como ves, hemos utilizado la función map para crear un array de componentes User. Cada componente tendrá un prop name y email, además de otro prop llamado key. Sin este prop la aplicación funcionará pero dará un warning al crear muchos componentes iguales sin clave identificativa.

PASO 3. LLAMANDO AL COMPONENTE USERLIST

Es hora de llamar al componente UserList desde nuestro componente App. Puedes observar el código que hemos cambiado porque está resaltado con un tipo de letra más grande.

```
import React, { Component } from 'react';
import UserList from './UserList';
import logo from './logo.svg';
import './App.css';
class App extends React.Component {
 constructor() {
   super();
```

```
    this.state = {
      users: [
        {id: 1, name: "perico", email: "perico@myfpschool.com"},
        {id: 2, name: "juanico", email: "juanico@myfpschool.com"},
        {id: 3, name: "andrés", email: "andrés@myfpschool.com"}
      ]
    };
  }
  render() {
    return (
      <div className="App">
        <div className="App-header">
          <img src={logo} className="App-logo" alt="logo" />
          <h2>Me mola Myfpschool</h2>
        </div>
        <div>
          <p><strong>Añade usuarios</strong></p>
              <UserList users={this.state.users}  />
        </div>
      </div>
    );
  }
}

export default App;
```

El cambio son solamente dos líneas. Una para importar el componente y otra para invocarlo dentro de la renderización.

PASO 4 CREANDO EL COMPONENTE USERFORM

Es hora de pensar en crear nuevos usuarios en nuestra aplicación. Para ello necesitaremos un componente que haga de formulario. A este componente le llamaremos UserForm.

Archivo UserForm.js

```
import React, { Component } from 'react'

class UserForm extends Component{
  render(){
```

```
    return (
      <form onSubmit={this.props.onAddUser}>
          <input type="text" placeholder="Nombre" name="name" />
          <input type="email" placeholder="Email" name="email" />
          <input type="submit" value="Guardar" />
      </form>
    );
  }
}
export default UserForm;
```

Como se puede observar, el formulario es un formulario normal salvo que la acción onSubmit está asociada a la función onAddUser que el componente padre de éste le va a enviar vía props.
En los siguientes apartados te explicaremos cómo asociar un evento de un elemento hijo a una función o método de un componente padre. A estas funciones se les suele denominar handlers.

PASO 5 CREANDO EL HANDLER

El handler será un método del componente App que maneje el formulario. Como podrás observar en el código siguiente, el método preventDefault() lo que hará será evitar que el navegador se refresque (acción por defecto) tras enviar el formulario.
Los datos del formulario estarán en event.target. De ahí obtendremos el nombre y el email del usuario a insertar.
El código de después es sencillo de comprender. Es simplemente modificar el estado (state).

```
 handleOnAddUser (event) {
    event.preventDefault();
    let user = {
      name: event.target.name.value,
      email: event.target.email.value
    };
    let tmp = this.state.users;
    tmp.push(user);
    this.setState({
      users: tmp
    });
  }
```

Importante: Para modificar el estado…

Tendrás que seguir los siguientes pasos:

1. Primero hacemos una copia del state a modificar.
2. Modificamos la copia con los datos que queremos.
3. Actualizamos el estado usando setState(). Si no lo hacemos así, los cambios no se renderizarán.

PASO 6 LLAMANDO AL COMPONENTE USERFORM

Ahora una vez creado el componente UserForm y programado su handler deberemos hacer dos cosas:

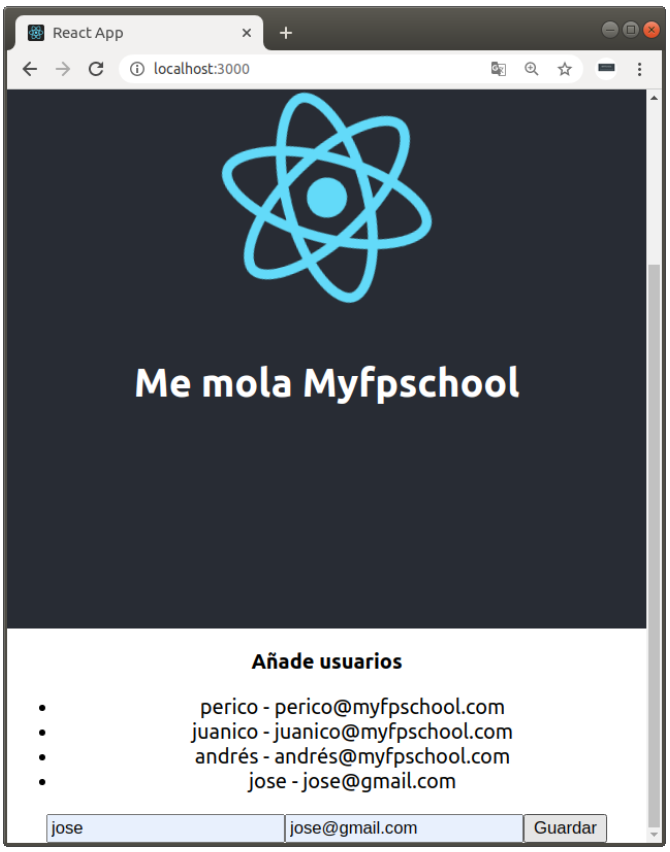
1. Importar el componente en el archivo App.js

```
import UserForm from './UserForm';
```

2. Realizar la llamada al mismo desde el render() del componente App pasándole como es obvio por props el handler que hemos creado.

```
<UserForm onAddUser={(e)=>this.handleOnAddUser(e)} />
```

6.1 Aspecto de la aplicación final



6.2 Código de la aplicación final (fichero App)

```
import React, { Component } from 'react';
import UserList from './UserList';
import UserForm from './UserForm';
import logo from './logo.svg';
import './App.css';
class App extends React.Component {
 constructor() {
   super();
   this.state = {
     users: [
       {id: 1, name: "perico", email: "perico@myfpschool.com"},
       {id: 2, name: "juanico", email: "juanico@myfpschool.com"},
       {id: 3, name: "andrés", email: "andrés@myfpschool.com"}
     ]
   };
 }
```

```
 handleOnAddUser (event) {
   event.preventDefault();
   let user = {
     name: event.target.name.value,
     email: event.target.email.value
   };
   let tmp = this.state.users;
   tmp.push(user);
   this.setState({
     users: tmp
   });
 }

 render() {
   return (
     <div className="App">
       <div className="App-header">
         <img src={logo} className="App-logo" alt="logo" />
         <h2>Me mola Myfpschool</h2>
       </div>
       <div>
         <p><strong>Añade usuarios</strong></p>
         <UserList users={this.state.users}  />
         <UserForm onAddUser={(e)=>this.handleOnAddUser(e)} />
       </div>
     </div>
   );
 }
}

export default App;
```